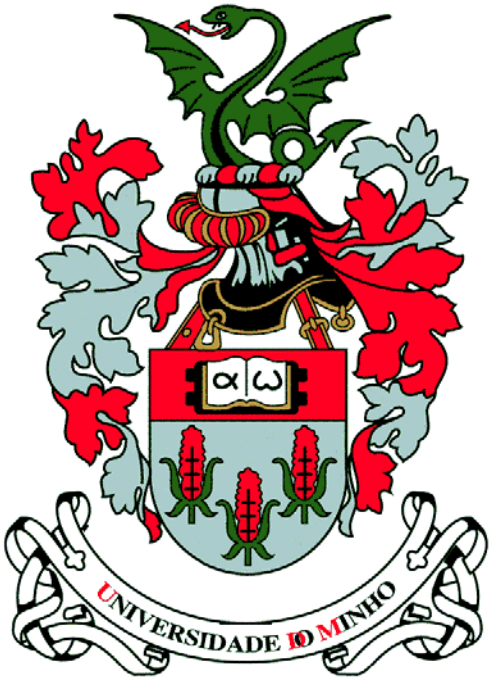
# Suporte à edição cooperativa de Informação Geográfica em ambiente *Web*

**Nuno André de Sampaio Faria**




*Dissertação submetida à Universidade do Minho para obtenção do grau de Mestre em Informática, elaborada sob a orientação de Jorge Gustavo Rocha*

Departamento de Informática
Escola de Engenharia
Universidade do Minho
Braga, 2006

## Resumo

A disseminação da Internet na sociedade preconiza uma revolução na divulgação de muitas áreas do conhecimento. Os Sistemas de Informação Geográfica (SIG) têm o potencial para ser uma dessas áreas. Com a distribuição de informação georeferenciada e procedimentos para a sua manipulação, através da *World Wide Web* (WWW), efectiva-se um serviço de difusão universal, ao facilitar o acesso aos dados a partir de qualquer ponto por qualquer utilizador. Tipicamente, a Informação Geográfica (IG) é capturada e tratada por empresas especializadas. O papel dos clientes resume-se, na maior parte dos casos, a meros consumidores dessa informação.

Neste trabalho, é proposta a utilização de formatos e tecnologias não proprietárias para a realização de uma plataforma em que os utilizadores podem enriquecer (acrescentando, actualizando e removendo IG) um determinado conjunto de dados geográficos. O esforço do *Open Geospatial Consortium, Inc.* (OGC) no que respeita à criação de mecanismos para disponibilizar e manipular IG, de uma forma livre e através da *Web*, tem sido enorme. Daí apareceram os Geo-WS, sendo de referir os casos de sucesso dos serviços de mapas e de entidades geográficas (WMS e WFS, respectivamente).

Para além das propostas do OGC, o foco desta dissertação passa por: compreender as especificidades inerentes à IG; compreender as maneiras de representar, guardar e disponibilizar IG; e contribuir para facilitar a apresentação e manipulação da IG na *Web*, através de exemplos concretos criados para o efeito (destaca-se o GeoWiki, uma aplicação *Web* criada de raiz, baseada no conceito da Wikipédia, para manipular IG).

Desta forma pensa-se ter conseguido demonstrar que a manipulação de IG via *Web* é um processo pouco dispendioso, há tecnologia para tal e é de total interesse por parte dos utilizadores da IG.

## Abstract

The spread of the internet throughout society has sparked a revolution in the way many areas of knowledge are promoted. Geographic Information Systems (GIS) is potentially such an area. By distributing geospatial information and procedures to manipulate it across the Web, a universal broadcast service is achieved, allowing users to access data anywhere. Geographic Information (GI) is usually the realm of specialized companies, and the client's role is often limited to consuming such information.

In this dissertation, we propose the use of non-proprietary formats and technologies to develop a framework in which the users themselves can manipulate a particular set of geographic data (adding, updating and removing GI). The Open Geospatial Consortium, Inc. (OGC) has striven greatly to create methods for manipulating GI and to make it freely available on the Web, prompting the rise of Geo-WS. Some have being widely adopted, like the map and feature services (WMS and WFS respectively).

In addition to the services proposed by the OGC, this dissertation also focuses on: understanding GI specifics; understanding different representations, GI storage and retrieval; and GI presentation and manipulation on the Web, through examples created for that purpose (with special emphasis on GeoWiki, a developed Wikipedia-inspired Web application for manipulating GI).

We were able to demonstrate that GI manipulation on the Web is an inexpensive process that uses available technology and is also of interest to the GI user community.

## Agradecimentos

Uma dissertação de mestrado não se conclui de um dia para o outro, é um processo longo, e muitas vezes difícil, que envolve esforço e dedicação. Durante esta jornada, muita gente colaborou comigo, me deu apoio e incentivo para conseguir superar as dificuldades.

Assim, agradeço ao Professor Pedro Henriques, meu primeiro orientador, pelo empenho e energia transmitidos, quer durante a pós-graduação quer no início da dissertação. Agradeço também a todos os meus actuais e ex-colegas de laboratório, que contribuíram para o bom ambiente de trabalho, sempre presente e essencial. Aos meus amigos, o meu obrigado pela força transmitida e pela paciência e compreensão sempre que os "troquei" para trabalhar nesta dissertação.

Agradeço a colaboração do Miguel Pinto, do Paulo Rocha e do Luís Rodrigues na revisão do resumo e respectiva tradução para inglês no *abstract*. À Susana Correia estou-lhe eternamente agradecido pela revisão geral do texto.

Para finalizar, agradeço, do fundo do coração, ao meu orientador e à minha família. Ao meu orientador, Jorge Rocha, pelos já longos anos de trabalho e cooperação mútua, pela camaradagem e amizade, pelo bom ambiente, pela ajuda na minha formação académica, pelo incentivo, força e dedicação a esta causa comum, o meu muito obrigado. À minha família, principalmente aos meus pais, pelo carinho, pela paciência, pela entrega e determinação na minha formação, dedico esta dissertação.

# Índice

## Índice de Figuras

# 1. Introdução

A partir do momento em que a Internet começou a fazer parte da vida quotidiana do cidadão comum, os computadores deixaram de ser apenas máquinas que processam informação para se tornarem, também, o meio para aceder e manipular toda e qualquer informação disponível (a qualquer hora, em qualquer lugar e para qualquer plataforma). A Internet possibilitou (e cada vez mais possibilita) ainda a partilha e troca de informação de um modo mais abrangente, para domínios gerais ou específicos e entre utilizadores ou aplicações [Kingstonetal03]. A preocupação com a forma como a informação é partilhada e trocada entre cada domínio específico, originou a que se desenvolvessem esforços para normalizar ao máximo essas mesmas trocas (para não se cair no ridículo de cada fornecedor de informação disponibilizá-la à sua própria maneira e cada cliente ser "obrigado" a perceber como é que cada fornecedor o faz). No domínio da Informação Geográfica (IG), cedo se percebeu a necessidade de normalizar as trocas de IG sobre a *Web*, conscientes dos insucessos do passado em lidar com formatos proprietários.

As potencialidades dos Sistemas de Informação Geográfica (SIG), só recentemente começaram a poder ser exploradas por um leque vasto de utilizadores, da arqueologia à biologia, do marketing à saúde. Nos anos 80 e 90, a sua massificação esteve sempre comprometida por factores ligados à própria tecnologia e às estratégias dos próprios fornecedores. Um SIG era sinónimo de um sistema complexo, fechado e proprietário, de níveis de desempenho insuficientes e de elevados investimentos [Klosterman01].

Mais recentemente, os SIG têm evoluído de sistemas fechados e especializados para sistemas abertos e de abrangência corporativa, ou transversais à empresa, capazes de suportar não só os processos de negócio das empresas, como a desempenhar um papel estratégico no suporte à decisão, logística e gestão da distribuição, gestão de obras e projectos e na gestão de activos [Craigetal02].

Neste contexto, a IG é cada vez mais utilizada no dia-a-dia, como componente essencial do processo de decisão. Com o desenvolvimento dos *Web Services* (WS) relacionados com a IG (em particular, resultado das iniciativas do OGC[1]), criou-se uma camada computacional que nos permite abstrair dos formatos, algoritmos e outras especificidades da IG. Com base nos serviços preconizados, é possível, por composição, integrar determinadas funções no software.

Enquanto que a vocação natural dos servidores é a distribuição de informação ou serviços, é intenção deste projecto usá-los como ponto de partida para a criação de repositórios de IG em ambiente cooperativo. Ou seja, pretende-se dar a possibilidade de vários utilizadores poderem construir, em comum, um determinado conjunto de dados.

Dado que uma das características da IG é que ela existe mais predominantemente junto ao local a que se refere [Davis03], este tipo de ambiente colaborativo será altamente distribuído geograficamente. Provavelmente, será mais

[1] http://www.opengeospatial.org/

fácil começar esta construção colaborativa em domínios muito específicos, em que se controla o tipo de dados e o seu significado. À medida que for mais aberto o domínio da aplicação (por exemplo, cadastrar todos os pontos negros ambientais do planeta) maior suporte terá que haver a modelos de dados menos rígidos (tabelas versus documentos XML) e melhores terão de ser as ferramentas relacionadas com o significado (ontologias multi-língua).

## *1.1. Motivação*

Até muito recentemente, era muito difícil actualizar a IG. Era uma área dominada por um conjunto muito limitado de profissionais e o mercado também se restringia a meia dúzia de empresas fornecedoras de software e de serviços. A própria produção de IG era muito restrita: os militares detinham a produção da maioria da informação utilizada em fins civis. O próprio suporte dominante, sobre o qual era distribuída e utilizada a IG, era o papel.

Com a queda dos preços dos equipamentos, do desenvolvimento de software, da abertura do sistema GPS a aplicações civis, e, acima de tudo, com o aparecimento de IG na *Web* em *sites* como o *Google Maps*[2], *Via Michelin*[3], A9[4], entre outros, os utilizadores começaram a exigir que as aplicações manipulassem também IG. Até os tradicionais mapas das estradas, em papel, deram lugar a sistemas de localização e encaminhamento de baixo custo, hoje bastante vulgarizados.

A metáfora em que se baseia este trabalho (manipulação de IG de um modo cooperativo) inspira-se na Wikipédia[5]. Nesta, qualquer utilizador pode contribuir para a informação lá contida, o que origina que a informação sobre qualquer tópico esteja frequentemente a ser actualizada, à medida que os acontecimentos estejam a ocorrer, numa dialéctica que tende a constituir uma síntese muito abrangente. A Wikipédia constituiu um marco muito importante na história da informação dado que permitiu, a grande escala, de um modo simples e distribuído, a criação de grandes repositórios de informação em que a manutenção é assegurada pela própria comunidade de utilizadores. Sendo que cada utilizador pode emitir a sua opinião, pode ser difícil manter a objectividade e credibilidade da informação, mas mesmo essas questões são ultrapassadas quase sempre com sucesso, devido à cooperação de todos os interessados.

Então, porque não usar os mesmos conceitos da Wikipédia, aplicados à produção ou actualização cooperativa de IG? Num cenário perfeito (e à escala local), num repositório da rede viária da cidade de Braga, qualquer utilizador que tenha conhecimento de uma nova rua criada ou de um sentido alterado (nem que seja temporariamente), pode assinalar essa mudança para que o repositório esteja sempre actualizado. Outros utilizadores poderão confirmar ou, pelo contrário, contradizer a informação introduzida e, recorrendo a algum mecanismo de acreditação dos utilizadores, chegar-se-á a um repositório sempre actualizado.

[2] http://maps.google.com/
[3] http://www.viamichelin.com/
[4] http://maps.a9.com/
[5] http://www.wikipedia.org/

Embora ainda não se possa, de todo, reclamar o sucesso desta abordagem, pois ainda não está vulgarizada na Internet, o que se pretende mostrar nesta dissertação é o suporte tecnológico desenvolvido, sob o nome GeoWiki.

## *1.2. Objectivos*

Este trabalho versa a manipulação de IG em computador, com o objectivo último de criar uma plataforma onde seja possível os utilizadores participarem na construção cooperativa de repositórios de IG, num ambiente *Web*.

Mas, para se atingir esse objectivo, é necessário assinalar uma mudança de mentalidade muito grande, rompendo-se com o paradigma enraizado de que é preciso uma produtora de IG (que tem de estar licenciada e ser-lhe concedido um alvará de exploração), uma outra entidade que a valida e, eventualmente, outras que a distribuem e comercializam. A cartografia, de alguma maneira, é muito exigente no rigor cartográfico, quando, em muitas aplicações, este pode ser preterido a favor do rigor topológico, por exemplo. Argumentamos que é muito mais importante, por exemplo, identificar o local aproximado de resíduos radioactivos abandonados, do que determinar com rigor o erro e os desvios com que esse registo foi efectuado.

Pretende-se mostrar que, assim como a restante informação é facilmente manipulada e actualizada via *Web*, por não profissionais, a IG também o poderá ser. Mostra-se, então, como é possível criar e manter conjuntos de dados geográficos, completamente via *Web*, de uma forma distribuída. As ferramentas de edição, em si mesmas, não são muito sofisticadas, pois a prioridade era prototipar todo o GeoWiki. Para domínios onde hajam restrições mais exigentes ao nível de edição, o ambiente gráfico terá que contemplar essas restrições (algumas são semânticas, e fazem sentido em domínios específicos).

Para isso, os três grandes objectivos desta dissertação passam, nomeadamente, por:

- Compreender as especificidades da IG;
- Conhecer e desenvolver soluções que manipulem a IG num ambiente *Web*;
- Contribuir para facilitar a apresentação e manipulação da IG na *Web*.

Ao nível tecnológico, a concretização do GeoWiki, lança um desafio bem interessante para armazenar e recuperar as múltiplas versões de cada entidade geográfica.

## *1.3. Organização da dissertação*

Esta dissertação encontra-se organizada em 6 capítulos que mostram a evolução e sequência deste trabalho para atingir os objectivos propostos. Assim, toda a descrição deste trabalho está enquadrada pela introdução (capítulo 1) e a conclusão (capítulo 6), com que se encerra, lançando também ideias para um trabalho futuro.

No capítulo 2, abordam-se os conceitos mais relevantes para a compreensão de algumas das especificidades da IG que são usadas neste trabalho. Também neste capítulo, mostram-se as tecnologias mais aceites e implementadas para a representação vectorial de IG, sejam elas proprietárias ou livres.

No capítulo 3, introduz-se o conceito de *Geo Web Services* (Geo-WS) e apresentam-se os serviços preconizados pelo OGC. Como este capítulo é dedicado à publicação de IG na *Web*, são mostradas implementações *open source* de servidores Geo-WS: o projecto *deegree* e o projecto *GeoServer* (este último usado para suporte aos casos de estudo apresentados). Ainda neste capítulo, são apresentadas formas alternativas (aos serviços do OGC) de publicação de IG na *Web* como é o caso da IG em SVG e a API do *Google Maps*.

No capítulo 4, é apresentado o caso de estudo, um *site* de turismo que inclui um SIG e que permite a visualização e manutenção da componente geográfica. Assim, neste capítulo mostra-se como configurar o *GeoServer* para disponibilizar IG e também são apresentados vários clientes para integração no *site* para visualizar IG.

O capítulo 5 é dedicado à actualização de IG na *Web* e vem no seguimento do capítulo anterior, pois as propostas apresentadas têm como base o caso de estudo. Aqui, são expostas as diferentes maneiras encontradas para satisfazer os requisitos do *site* (actualização de coordenadas). Neste capítulo, apresenta-se o GeoWiki, que é a principal proposta deste trabalho de mestrado. Concretamente, demonstra-se uma aplicação via *Web* para actualização e manutenção de rede viária, com controlo de versões.

# 2. A Informação Geográfica

A IG, no seu todo, denota certas especificidades que convém ressalvar e compreender. Vários são os autores que defendem que a IG deve ser tratada de uma forma particular no contexto das ciências da computação (CC), sendo que Rocha [Rocha05], na sua dissertação, trata exaustivamente essa vertente.

Este capítulo fala das especificidades inerentes à IG e está dividido em duas partes. Na primeira, dá-se a conhecer um pouco mais sobre as várias ciências (ditas de geociências) e tecnologias que permitem recolher, representar e processar IG, nomeadamente para a criação de mapas em formato digital. Na segunda parte, mostram-se várias estruturas e tipos de dados que permitem guardar e representar IG.

## *2.1. Especificidades e Conceitos*

Designa-se por **Informação Geográfica** (IG) a informação acerca de entidades ou fenómenos localizados na proximidade da superfície terrestre [Rocha05]. Ou seja, de um modo abstracto, pode-se pensar em IG como algo (de relativa importância) que ocorre algures na superfície da Terra. Então, daqui pode-se inferir que a IG é um par:

$$IG = (Atributos \times Localização)$$

Para um conjunto de atributos (ou informação relacionada) de uma instância de um objecto é associada uma localização efectiva. A localização está associada a um sistema de referência onde a mesma faça sentido. A localização pode ser dada por um topónimo (Lixa) ou, de uma forma mais matemática, pode ser dada por um tuplo que contém as coordenadas geográficas e a altitude, se esta se revelar de interesse.

Partindo deste conceito de IG, pode-se agora pensar, de um modo resumido, em como é que é feito o levantamento de IG, nomeadamente como é possível identificar a localização dos objectos à face da Terra.

Todos sabemos que, embora a Terra seja quase esférica, a sua forma real é bastante complexa, apresentando muitas deformações relativamente a um elipsóide. No entanto, existe uma ciência que trata e pensa nestas questões:

**Geodesia** – Ciência que se ocupa do estudo da forma e dimensões da Terra, bem como das suas deformações e movimentos [Caviedes05].

A Geodesia trata a figura da Terra como um Geóide[6]. A determinação da posição de pontos sobre a superfície da Terra requer a definição de sistemas de referência (o mais usado actualmente é o WGS84, que é um *datum* de origem geocêntrica de referência para a utilização de Sistemas de Posicionamento Global - GSP) e para tal é necessário conhecer a forma real do nosso planeta. Nesse contexto,

[6] Modelo físico da forma da Terra ou, segundo Gauss, a figura matemática da Terra

existem, em todos os países, pontos cujas coordenadas (latitude, longitude e altitude) foram calculadas com grande precisão, através de técnicas diversas, entre as quais o GPS. Estes pontos estão marcados fisicamente no terreno através de marcos e denominam-se de **vértices geodésicos** [Gemael99].

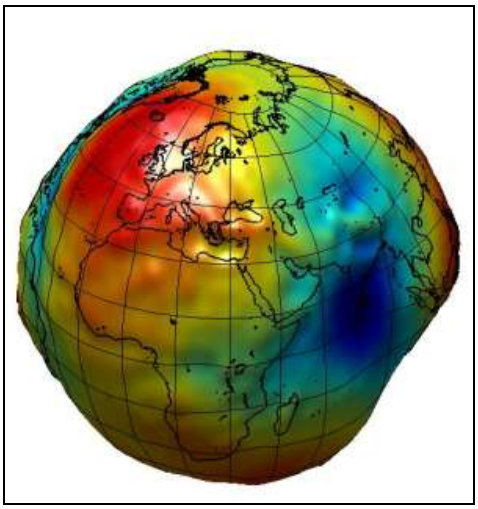


**Figura 1 - Geóide como representação da Terra.**

É certo que, para muitas aplicações (as descritas nesta dissertação são exemplo disso), não é necessária tanta complexidade para representar e comunicar a IG de interesse e, nesses casos, é suficiente utilizar uma superfície mais simples e bem adaptada a cada localização específica. Uma forma de fazer isso é representar a IG sobre uma superfície plana, como é o caso de mapas em papel. Aqui recorre-se a uma outra ciência:

**Cartografia** – Ciência que trata da concepção, produção, difusão, utilização e estudo dos mapas[7].

Como a Terra não é plana, é necessário desenvolver formas de a representar num plano tornando-se esta representação mais fácil de imprimir, transportar, etc., mesmo com algum prejuízo em termos de rigor. Isto consegue-se projectando os vários pontos da superfície terrestre sobre uma superfície que possa ser facilmente transformada num plano, como é o caso de um cilindro ou de um cone [Câmaraetal04].

[7] A Associação Cartográfica Internacional (ACI) adopta a seguinte definição para Cartografia: "Conjunto dos estudos e operações científicas, técnicas e artísticas que intervêm na elaboração dos mapas a partir dos resultados das observações directas ou da exploração da documentação, bem como da sua utilização"

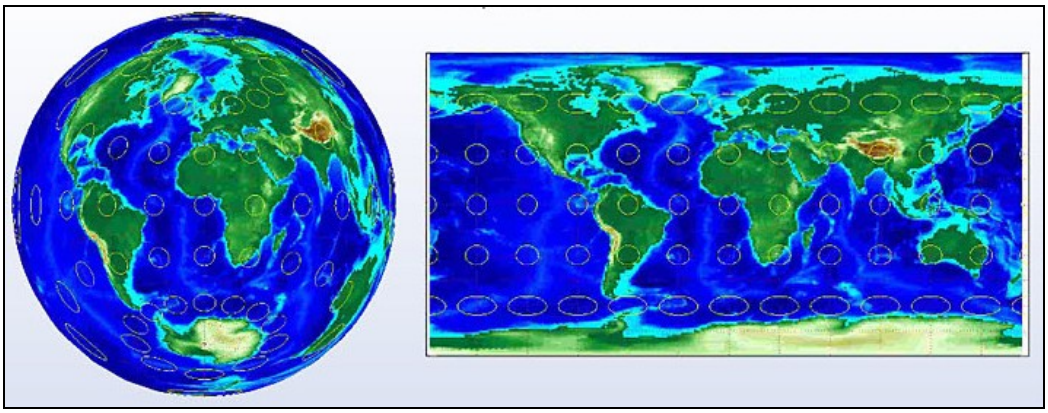


**Figura 2 - Diferentes projecções para representar diferentes zonas da Terra.**

Como é óbvio, estas projecções introduzem deformações na sua representação, mas também não deixa de ser verdade que, se se conhecer a forma como é feita cada projecção, pode-se determinar as deformações provocadas, o que permite escolher a projecção adequada em função do objectivo do mapa a construir (mapas de estradas, fronteiras, de temperaturas ou topográficos, etc.).

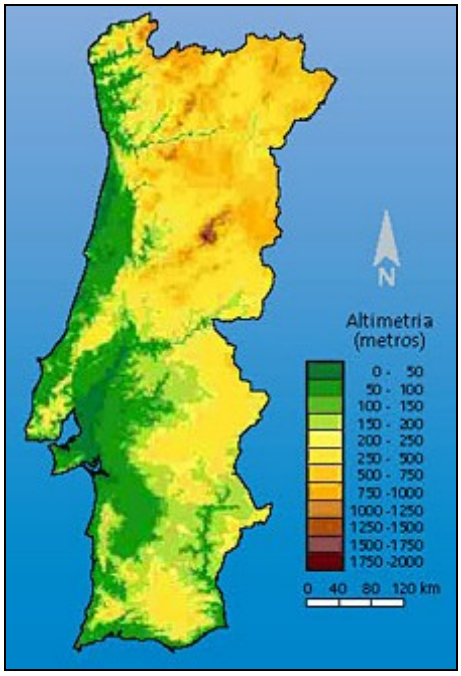



**Figura 3 - Mapa de altitude de Portugal.**

Uma outra ciência que contribui para esta temática é a:

**Topografia** – Ciência que se ocupa da determinação da posição de pontos situados sobre a superfície terrestre, utilizando aparelhos que medem distâncias e ângulos[8].

[8] A Associação Nacional de Topógrafos (ANT) define-a como a ciência que estuda os instrumentos, métodos de operação no terreno, cálculos e desenhos necessários ao levantamento e representação gráfica, mais ou menos detalhada de uma parte da superfície terrestre.

É importante o estudo de todos os acidentes geográficos para definir a situação e a localização de uma área em geral (relativamente pequena, dado que para toda a superfície terrestre e/ou grandes áreas geográficas tem-se, como já visto, a geodesia).

Em topografia definem-se as medidas de área, distância, ângulos e nivelamento e volumes. Na maioria das utilizações não profissionais, a terra até pode ser considerada plana, pois a distância entre os vários pontos de interesse não é muito grande. No entanto, podem existir casos em que seja necessário construir mapas topográficos que mostrem a elevação do terreno relativamente ao nível do mar. Isso pode ser feito usando linhas que conectem pontos com a mesma elevação (ou cota), denominadas de **curvas de nível**[9].

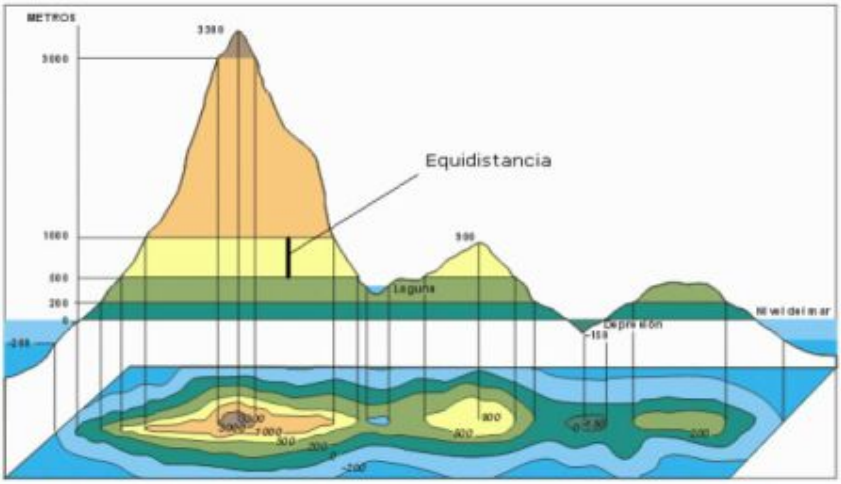



**Figura 4 - Representação do relevo usando curvas de nível.**

Uma técnica muito usada na construção de cartas (mapas) topográficas é a **fotogrametria** que consiste na medição rigorosa de pontos (locais) a partir de fotografia. O objectivo é a identificação de objectos ou fenómenos em fotografias e determinar a sua forma e posição exactas [Temba00]. As fotografias podem ser tiradas de um avião ou satélite (fotografias aéreas) ou de um local à superfície da Terra (fotografias terrestres) e, para que seja possível determinar a posição dos objectos nas fotografias, é necessário que estes estejam representados em duas fotografias tiradas de pontos diferentes. A sobreposição dessas fotografias, colocadas exactamente na posição relativa que ocupavam no instante em que foram tiradas, vai originar, devido a diferentes perspectivas de um mesmo local, uma percepção das três dimensões do terreno, permitindo assim medir altitudes de forma rigorosa. Esse feito é denominado por **estereoscopia** [Temba00].

[9] http://pt.wikipedia.org/wiki/Topografia

**Figura 5 - Campus de Gualtar visto pelo *Google Maps* (usa a técnica da fotogrametria para produção dos mapas de satélite).**

Paralelamente à técnica da fotogrametria, pode-se usar o conhecimento da posição geográfica e a altitude de um conjunto de pontos do terreno para a construção (naturalmente em computador) de um modelo de relevo da zona onde esses pontos estão situados. Estes modelos são designados por **Modelos Digitais do Terreno** e permitem conhecer o valor da altitude de todos os pontos do terreno, mesmo daqueles em que esta não foi medida [MüllerGarcia01]. Projectando imagens obtidas pela técnica da fotogrametria sobre os modelos digitais do terreno, obtêm-se mapas mais realistas permitindo, para além disso, e entre outras coisas, calcular declives e exposições de terrenos e determinar as zonas que são visíveis a partir de um determinado ponto.

**Figura 6 - Modelação Digital do Terreno (Campus de Gualtar segundo a aplicação *Google Earth*[10] mostrando a inclinação do terreno)**

Todas estas ciências foram apresentadas para se chegar aos **Sistemas de Informação Geográfica** (SIG), que são sistemas computacionais onde é armazenada e manipulada a informação acerca do que existe à face da Terra e dos fenómenos que aí ocorrem [Martins04]. O objectivo é a criação, em computador, de modelos da realidade que, baseados, entre outras, nas técnicas anteriormente descritas, permitem e facilitem a análise, gestão ou representação do espaço e dos fenómenos que nele ocorrem para serem aplicados a diversos fins (determinar caminhos mais curtos entre dois lugares, delimitar as zonas históricas de uma cidade, prever os efeitos de um furacão, suportar a tomada de decisões face a uma ameaça de pandemia, etc.). Modelar, por exemplo, a evolução de um incêndio florestal, obriga não só a conhecer toda a componente geomorfológica, mas também a componente meteorológica e o seu comportamento, e também os modelos de combustão do coberto vegetal. Por isso, os SIG são muito mais que meras representações em computador; têm a capacidade de modelar os processos e os fenómenos que ocorrem na superfície da terra ou na sua proximidade.

## *2.2. Representação Vectorial da IG*

A sociedade vive uma crescente procura por sistemas para armazenamento de informações, visando organizar uma produção de conhecimento cada vez maior. Quando a cartografia entrou na era digital, a necessidade de armazenamento também se tornou essencial, pois as bases evoluíram agregando uma quantidade cada vez maior de informação para atender a novos imperativos do mercado.

A informação armazenada pode estar sob a forma de imagens, tabelas, ficheiros de texto, etc., sendo que nesta dissertação é dada especial atenção à representação vectorial de IG. Neste modelo, o foco das representações centra-se na

[10] http://earth.google.com

utilização de elementos matemáticos, relativamente posicionados a um sistema de eixos, para modelar a localização das entidades. Utilizam-se essencialmente três elementos: o ponto, a linha e o polígono.

Embora esta secção se refira exclusivamente à representação vectorial, convém referir que existem também representações matriciais (ou *raster*) que compartimentam o espaço em células regulares (habitualmente quadradas, mas podendo ser rectangulares, triangulares ou hexagonais). Cada célula representa um único valor. Quanto maior for a dimensão de cada célula (resolução) menor é a precisão ou detalhe na representação do espaço geográfico. Em GML, uma forma mais recente de codificação e modelação de IG, é possível representar informação quer vectorialmente, quer matricialmente. Pode-se, inclusivamente, em GML, associar à mesma entidade, mais do que uma representação da localização. Por exemplo, uma cidade pode ser representada por um polígono e por um ponto, podendo-se optar por uma ou por outra, consoante as necessidades da análise ou da apresentação (por exemplo, em função da escala). Esta codificação é apresentada na secção 2.2.4.

Centrado na representação vectorial, seguidamente dão-se a conhecer as codificações mais utilizadas pelas ferramentas mais conhecidas.

### 2.2.1. *Shapefiles*

Uma *shapefile* é uma representação utilizada em muitos produtos de software SIG. Foi criado, em finais de 1997, pela ESRI[11] para utilização no produto *ArcView*[12].

Uma *shapefile* é um formato vectorial que guarda localizações geométricas (não topológicas) e informação sobre atributos associados [ESRI98]. A geometria para uma entidade[13] geográfica é guardada sob a forma de um conjunto de vectores de coordenadas (de valores reais). As *shapefiles* lidam com entidades geográficas singulares. As entidades geográficas suportadas incluem pontos, linhas e áreas (sendo estas representadas por polígonos fechados). A cada entidade está associado um conjunto de atributos guardados no formato dBASE.

Devido ao facto de ser um formato vectorial sem guardar as relações topológicas, por exemplo, as *shapefiles* têm certas vantagens sobre outras codificações, tais como: uma maior rapidez para as aplicações desenharem a informação que contêm, ocupam menos espaço em disco e a capacidade de serem facilmente editáveis (leitura e escrita).

São usados vários ficheiros para construir uma *shapefile*, sendo três o número mínimo, com as seguintes extensões:

- **.shp –** guarda as entidades geométricas;
- **.shx –** guarda os índices das entidades geométricas;

[11] Environmental Systems Research Institute, Inc. http://www.esri.com.
[12] http://www.esri.com/software/arcview
[13] Do inglês *feature* e, segundo Rocha [Rocha05], é a unidade mais básica de IG que se pode discriminar.

- **.dbf –** o ficheiro dBASE que guarda a informação sobre os atributos das entidades.

Para além destes ficheiros obrigatórios, outros podem ser incluídos para melhorar a performance das operações que inquerem a *shapefile*, para guardar informação sobre as projecções geográficas ou para guardar metadados. Os ficheiros opcionais são:

- **.sbn** ou **.sbx –** guarda o índice espacial das entidades;
- **.fbn** ou **.fbx –** guarda o índice espacial das entidades para *shapefiles* apenas de leitura;
- **.ain** ou **.aih –** guarda o índice do atributo dos campos activos de uma tabela ou uma tabela de atributos temáticos;
- **.prj –** guarda a informação sobre o sistema de coordenadas;
- **.shp.xml –** metadados da *shapefile.*

Esta codificação está descrita em pormenor no repositório[14] da ESRI. Muitas, senão a totalidade, das ferramentas disponíveis, comerciais e *open source*, lêem e escrevem este formato. Uma codificação *open source* que lê e escreve o formato *shapefile* está disponível em http://www.nrdb.co.uk/nrdbview/.

### 2.2.2. *Geodatabases*

*Geodatabases*[15] é o nome dado a um modelo de dados introduzido com o ArcGIS[16] [ESRI00], também da ESRI. O conceito, contudo, não é exclusivo da ESRI e surgiu com vista a aumentar as capacidades das bases de dados relacionais existentes com o suporte a dados geográficos. Na terminologia da *Intergraph*[17], estas bases de dados são designadas por *warehouses* e, no *Geomedia*[18], passou a ser possível juntar na mesma tabela entidades com representações espaciais diversas (pode coexistir uma entidade representada por um ponto e outra como uma área, por exemplo), embora cada entidade só tenha uma única representação espacial.

Basicamente, são bases de dados relacionais que contêm informação geográfica organizada numa hierarquia própria. Existem entidades que, por partilharem as mesmas propriedades, são agrupadas sob forma de tabela numa classe que as caracterize. Essas classes denominam-se classes de entidades (do inglês *feature classes*) e correspondem, grosso modo, às *shapefiles*. As classes que partilhem determinadas características comuns podem-se agrupar em conjuntos (denominados conjuntos de dados de entidades, do inglês *features datasets*).

[14] http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf
[15] http://www.esri.com/software/arcgis/geodatabase/index.html
[16] ArcGIS é uma colecção integrada de produtos SIG proprietária da ESRI. http://www.esri.com/software/arcgis/index.html
[17] http://www.intergraph.com/
[18] http://www.intergraph.com/geomedia/

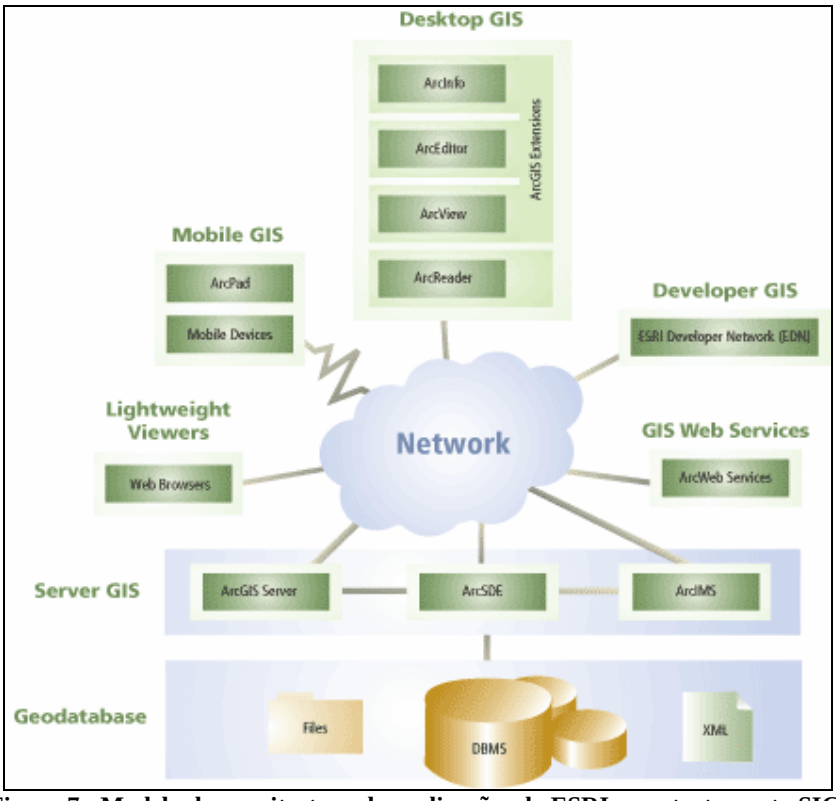



**Figura 7 - Modelo de arquitectura das aplicações da ESRI para tratamento SIG.**

As *Geodatabases* são bastante usadas em contextos SIG. Por um lado, exploram as vantagens dos SGBD, da mesma forma que os sistemas tradicionais o fazem, e além disso, porque têm mais algumas características especialmente desenvolvidas para suportar a componente geográfica. Destaca-se, a título de exemplo, a capacidade para:

- Introdução de dados mais exactos. Poucos erros são introduzidos, dado que, a maioria podem ser prevenidos por meio de regras de validação. Essas regras assentam num sistema de tipos mais extenso e mais robusto, com cláusulas de integridade referencial, definição de domínios mais restritos e regras de validação no próprio SGBD que impede a inserção de determinados valores;
- Organização hierárquica, agrupando classes, de forma a aumentar capacidade da *geodatabase* para representar o mundo real (por exemplo, os SGBD tradicionais não permitem agrupar as tabelas sob um determinado critério do analista);
- Armazenamento de pontos, linhas e polígonos num único campo *shape* e, além disso, todas as componentes que constituem a *geodatabase* (índices, tabelas, hierarquias, etc.) são encapsuladas num único ficheiro;
- Permitem, também, a definição e imposição de determinadas regras topológicas, o que representa um nível muito sofisticado de validação (pode-se obrigar a que nenhuma rua intercepte uma habitação).

Existem dois tipos de *Geodatabases*: *multiuser Geodatabases* e *single-user* ou *personal Geodatabases*.

### 2.2.2.1. Organização de uma *Geodatabase*

As tabelas das *Geodatabases* são formadas por linhas e colunas (tal como uma tabela de um SGBD relacional); as classes de entidades descrevem as entidades e os conjuntos de dados de entidades armazenam colecções de classes de entidades.

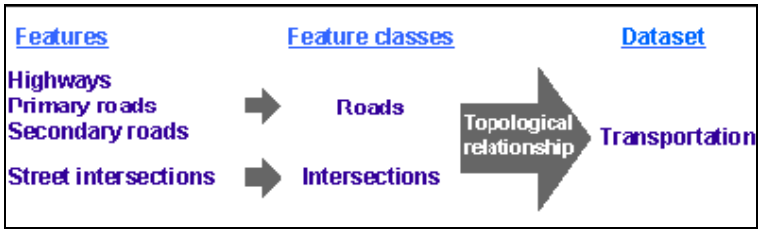



**Figura 8 – Exemplo de tipos de uma *Geodatabase*.**

No exemplo acima, todo o tipo de estradas (*Highways*, *Primary roads* e *Secondary Roads*) vai para dentro de *Roads* e aí deverá haver um campo que diga de que tipo é cada uma das vias lá representadas. *Roads* e *Intersections* foram agrupados num conjunto de dados de entidades denominado *Transportation*. Geralmente as classes de entidades dentro de um conjunto de dados de entidades possuem uma extensão geográfica similar, quer dizer que ocupam o mesmo espaço geográfico e podem ter relacionamentos topológicos.

Numa *Geodatabase* cada entidade é armazenada numa linha da tabela. O campo *shape* contém a geometria de cada entidade na tabela. As tabelas podem conter atributos adicionais para uma classe de entidade, tal como endereços, coordenadas e outras características.

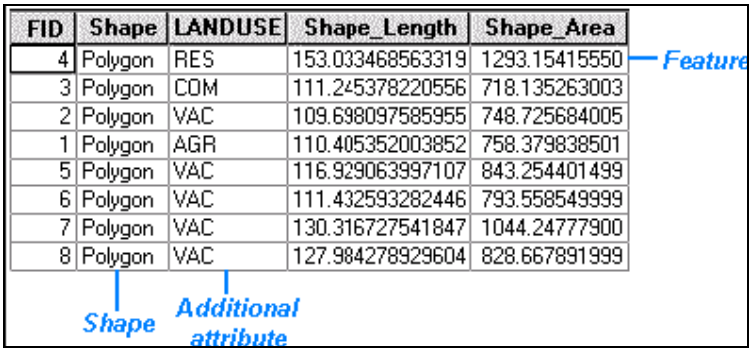



**Figura 9 - Tabela de entidades de uma *Geodatabase*.**

As classes de entidades armazenam entidades geográficas e os seus atributos, e podem também armazenar anotações. São formadas por entidades com um destes tipos de geometria: ponto, multi-ponto, linha ou polígono. Uma classe permite agrupar entidades homogéneas numa unidade. Por exemplo, estradas, ruas primárias e ruas secundárias, podem ser agrupadas dentro de uma classe denominada Transporte.

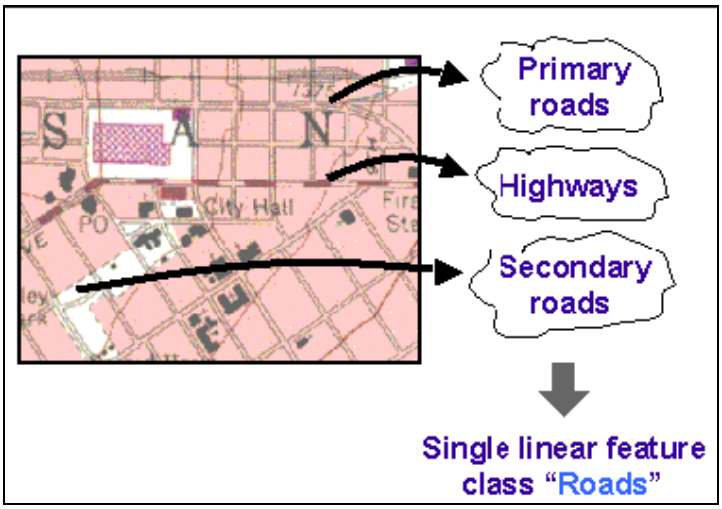



**Figura 10 - Uma classe de entidade como grupo homogéneo de objectos abstractos.**

A cada entidade individual numa classe é atribuído um identificador numérico e é caracterizada por localização (campo *shape*) e um único registo correspondente na tabela de atributos. O nome do identificador numérico pode variar de acordo com o formato dos dados, mas é importante salientar que existe uma relação de um-para-um entre entidade, identificador e atributos.

O utilizador pode agrupar classes de entidades numa unidade maior denominada conjunto de dados de entidades. Todas as classes num conjunto de dados de entidades compartilham a referência espacial (sistema de coordenadas).

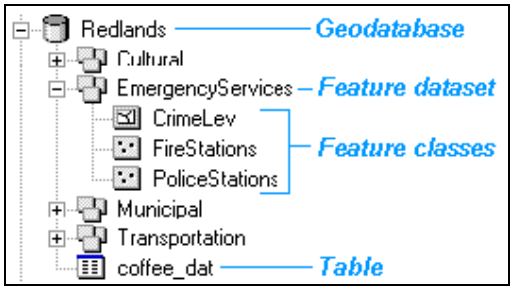



**Figura 11 - Vista das relações de uma *Geodatabase***

Por exemplo, no diagrama em cima, todas as classes de entidades no conjunto de dados de entidades *EmergencyServices* compartilham a referência espacial. A classe de entidades *FireStations* armazena entidades com geometria de pontos. O ícone visualizado do lado da classe de entidades identifica o tipo de geometria.

## 2.2.2.2. Validação de dados em *Geodatabases*

Em *Geodatabases* existe a noção de topologia – relações espaciais entre entidades geográficas – e a possibilidade de criar relações topológicas[19] para assegurar a qualidade dos dados.

[19] Subconjunto das relações espaciais caracterizado pela propriedade de se manter preservado sob transformações topológicas, tais como translações, rotações e escalonamento [Clementinietal93].

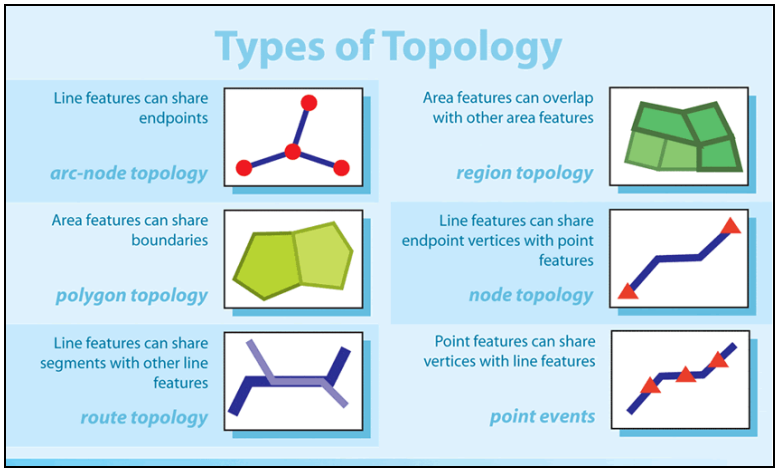



**Figura 12 – Tipos de topologia usados em *Geodatabases*.**

Nesse contexto, o uso de uma *Geodatabase* permite efectuar operações de validação. Validação significa impor restrições de edição, visualização ou análise, sobre determinadas entidades geográficas, dependendo das circunstâncias definidas pelo analista. As validações podem ser de dois tipos: espaciais ou de atributos.

A validação espacial pode ser usada de duas maneiras: pela definição de regras topológicas ou de redes geométricas. Para utilização de ambas, é preciso definir um conjunto de dados de entidades, onde deverão estar as classes de entidades que participarão da construção das regras. Uma mesma classe de entidades não pode participar simultaneamente na construção de duas regras. As regras topológicas aplicam-se a entidades como pontos, linhas e polígonos. As redes geométricas podem ser construídas combinando as classes de entidades linhas e pontos numa única entidade na *geodatabase* para modelar sistemas de redes, mantendo-se as relações topológicas entre as classes envolvidas.

Exemplos de regras topológicas:

- Dois polígonos distintos, da mesma classe, por exemplo, dois lotes, não podem ter partes sobrepostas;
- Pontos estarem obrigatoriamente dentro de um polígono. Este exemplo pode ser aplicado a um conjunto de cidades que pertencem a um país ou uma região;
- Uma linha que não se pode intersectar com ela própria. Este exemplo pode ser aplicado às curvas de nível ou linhas de contorno.

Exemplos de redes geométricas:

- Rede de metro e respectivas estações (as vias representadas por linhas e as estações por pontos);
- Rede eléctrica (os postes representados por pontos e as ligações entre eles por linhas).

A validação de atributos pode ser de três tipos: subtipos, domínios e classes de relacionamento. Estes tipos são necessariamente utilizados com dados tabulares, sendo necessários para manutenção da integridade dos dados e eficiência durante operações de gestão, visualização e edição. Um exemplo é quando se sabe à priori,

para um determinado campo, quais os possíveis valores que o mesmo poderá ter ou quando se querer validar os dados de um campo (por exemplo, para um campo que representasse a altura de postes, cujos valores devem estar entre 1 e 3 metros, definir-se-ia um domínio para este campo como > 1 (maior do que 1) e <= 3 (menor ou igual a 3)).

### 2.2.2.3. *Personal* vs *Multiuser Geodatabases*

Como já foi dito, existem dois tipos básicos de *Geodatabases*: *personal Geodatabases* e *multiuser Geodatabases*.

Uma *personal Geodatabase* é armazenada num ficheiro "*.mdb*" (formato usado para Microsoft Access) local e é apropriada para o uso pessoal ou para pequenos grupos de trabalho.

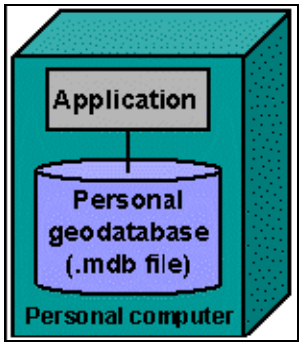



**Figura 13 – *Personal Geodatabase*.**

Uma *multiuser Geodatabase* está armazenada num servidor e é acedida através de aplicações como ArcSDE[20]. Os utilizadores devem-se ligar ao servidor para ter acesso aos dados que são administrados de uma forma centralizada e requerem um administrador de sistema para permissões e optimizações. Suportam múltiplas versões, o que permite que múltiplos utilizadores possam ver e editar os dados geográficos ao mesmo tempo, impondo regras para resolver conflitos de edição.

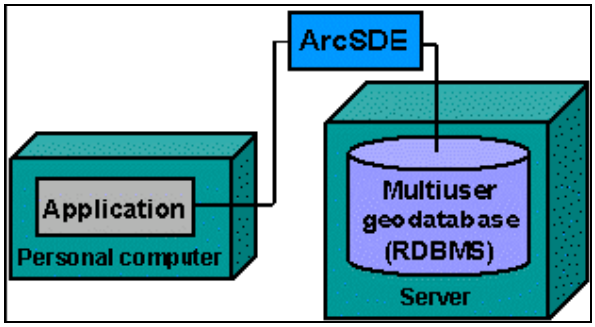



**Figura 14 – *Multiuser Geodatabase*.**

[20] http://www.esri.com/software/arcgis/arcsde/index.html

Uma *personal Geodatabase* possui toda a funcionalidade de uma *multiuser Geodatabase*, excepto a possibilidade de gerar versões (que é um processo apenas reservado às *multiuser Geodatabases*).

### 2.2.3. *Simple Features Specification* para SQL

Acompanhando as necessidades dos utilizadores de SIG, os Sistemas de Gestão de Bases de Dados (SGBD) criaram módulos (extensões) específicos para armazenamento e análise dos dados geográficos. Com isso, tornou-se possível organizar ambientes contendo sistemas clientes que acedam a todos os dados espaciais numa base de dados centralizada, tanto num servidor como num *cluster*. No entanto, para "impedir" que cada SGBD criasse a sua maneira de armazenar e aceder à IG, foi necessário criar normas.

As normas OpenGIS (*Open Geodata Interoperability Specification*), que têm vindo a ser desenvolvidas pelo OGC, resultam de um trabalho que tem como objectivo principal a criação de especificações (sejam de interfaces ou de codificações) que permitam a interoperacionalidade[21] no processamento de dados geográficos.

Estas especificações OpenGIS fornecem uma plataforma de trabalho, sobre a qual os programadores podem desenvolver software que permite aos seus utilizadores aceder e processar dados geográficos provenientes de várias fontes, através de uma interface de computação genérica, para uma base de informação tecnológica aberta [Buehleretal96]. No próximo capítulo, pág. 25, serão retomadas as especificações deste consórcio.

A concretização do OpenGIS para SQL [OGC99049], designada *Simple Feature Specification for SQL* (SFS), define um esquema SQL normalizado que suporta o armazenamento, consulta e actualização de colecções de dados geográficos simples usando a API do ODBC (*Open Database Connectivity*). Dados geográficos simples têm simultaneamente atributos espaciais e não espaciais. Atributos espaciais simples têm geometria 2D com interpolação linear entre os vértices. Estes dados são armazenadas em bases de dados relacionais, sendo cada atributo não espacial mapeado nos tipos de dados normalizados do ODBC/SQL92, e cada atributo espacial mapeado num valor geométrico, introduzido por esta norma.

Uma tabela cujas linhas representem entidades OpenGIS denomina-se tabela de entidades. Estas tabelas contêm uma ou mais colunas com valores geométricos. Na especificação OpenGIS descrevem-se implementações de tabelas de entidades para dois possíveis ambientes SQL: o SQL92 e o SQL92 com tipos geométricos. Ambos podem ser acedidos através do ODBC.

[21] Capacidade de comunicar, executar programas, ou transferir dados entre diferentes unidades funcionais sem que o utilizador tenha que se preocupar com as características específicas de cada uma dessas unidades [Rocha05].

Neste contexto, alguns SGBD permitem o tratamento de dados geográficos, destacando-se naturalmente as soluções proprietárias da IBM® (*DB2 Spatial Extender*) e Oracle® (através do *Oracle Spacial*).

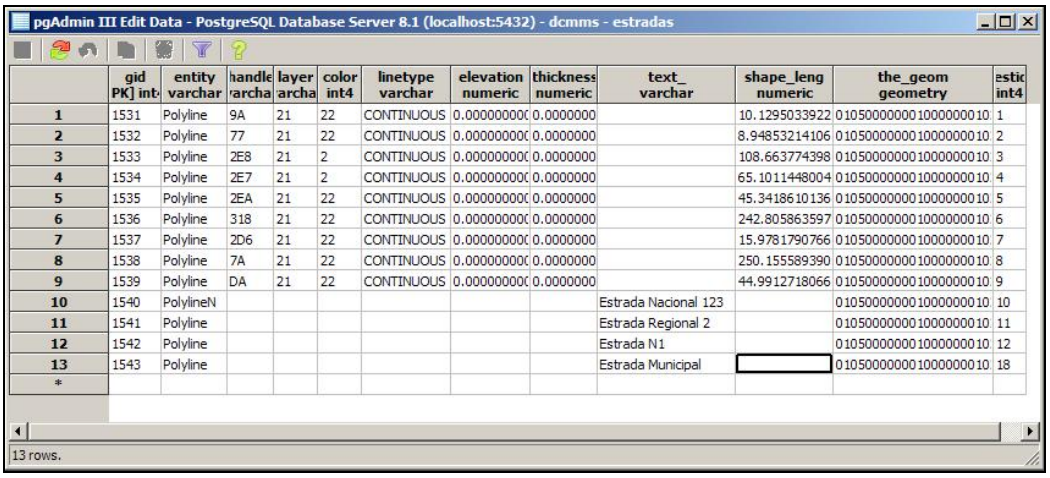
- ***Oracle Spatial*** – é uma extensão espacial desenvolvida sobre o modelo objecto-relacional do SGDB Oracle [Murray03]. Este modelo permite definir novos tipos de dados através da linguagem de definição de dados SQL DDL e implementar operações sobre esses novos tipos, através da linguagem PL/SQL [Urman02], uma extensão do SQL [Lassen98]. Esta extensão é baseada nas especificações do OpenGIS e contém um conjunto de funcionalidades e procedimentos que permitem armazenar, aceder, modificar e consultar dados espaciais de representação vectorial. O *Oracle Spatial* é formado pelos seguintes componentes: um modelo próprio de dados chamado MDSYS que define a forma de armazenamento, a sintaxe e semântica dos tipos espaciais suportados; mecanismo de indexação espacial; um conjunto de operadores e funções para representar consultas, junção espacial e outras operações de análise espacial; aplicações administrativas.

- ***DB2 Spatial Extender*** - o *DB2 Spatial Extender* [IbmDb2Ref] segue a norma SFS e permite armazenar informações geográficas em bases de dados DB2 para criar um sistema de informações geográficas: um complexo conjunto de objectos, dados e aplicações utilizados para gerar e analisar informações espaciais sobre recursos geográficos. O *DB2 Geodetic Extender* trata a Terra como um globo sem fronteiras ou junções nos pólos ou linhas de data. Utilizando os mesmos tipos de dados e funções espaciais de quaisquer outras operações do *Spatial Extender*, é possível utilizá-lo para executar consultas ininterruptas de dados transpolares e translineares de data. Os cálculos de distância e de área são precisos, independentemente da localização.

Mas, para caso de estudo desta dissertação, interessava, particularmente, sistemas não proprietários e, nesse sentido, existem soluções igualmente válidas neste campo. Assim, foram estudadas e testadas duas soluções que poderiam servir:

- ***Postgre*SQL** - O *PostgreSQL*[22] foi o primeiro SGBD de código aberto a trabalhar com um módulo específico para o tratamento dos dados geográficos vectoriais. Este módulo denominado de *PostGIS*[23] foi desenvolvido por uma empresa canadiana chamada *Refractions Research* e segue a especificação SFS do OGC. Para que o *PostGIS* contemple toda a SFS, é necessário que ele seja compilado juntamente com a biblioteca GEOS (*Geometry Engine OpenSource*). Com isso, o *PostGIS* passa a possuir mais de 130 funções e operadores para o tratamento de dados geográficos vectoriais, podendo atender todas as necessidades presentes numa instituição. Esta é uma solução já bastante madura e muito poderosa [UchoaFerreira04].

[22] http://www.postgres.org
[23] http://postgis.refractions.net




| | gid PK] int4 | entity varchar | handle varcha | layer varcha | color int4 | linetype varchar | elevation numeric | thickness numeric | text_ varchar | shape_leng numeric | the_geom geometry | estic int4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1531 | Polyline | 9A | 21 | 22 | CONTINUOUS | 0.000000000( | 0.0000000 | | 10.1295033922 | 0105000000100000001 | 1 |
| 2 | 1532 | Polyline | 77 | 21 | 22 | CONTINUOUS | 0.000000000( | 0.0000000 | | 8.94853214106 | 0105000000100000001 | 2 |
| 3 | 1533 | Polyline | 2E8 | 21 | 2 | CONTINUOUS | 0.000000000( | 0.0000000 | | 108.663774398 | 0105000000100000001 | 3 |
| 4 | 1534 | Polyline | 2E7 | 21 | 2 | CONTINUOUS | 0.000000000( | 0.0000000 | | 65.1011448004 | 0105000000100000001 | 4 |
| 5 | 1535 | Polyline | 2EA | 21 | 22 | CONTINUOUS | 0.000000000( | 0.0000000 | | 45.3418610136 | 0105000000100000001 | 5 |
| 6 | 1536 | Polyline | 318 | 21 | 22 | CONTINUOUS | 0.000000000( | 0.0000000 | | 242.805863597 | 0105000000100000001 | 6 |
| 7 | 1537 | Polyline | 2D6 | 21 | 22 | CONTINUOUS | 0.000000000( | 0.0000000 | | 15.9781790766 | 0105000000100000001 | 7 |
| 8 | 1538 | Polyline | 7A | 21 | 22 | CONTINUOUS | 0.000000000( | 0.0000000 | | 250.155589390 | 0105000000100000001 | 8 |
| 9 | 1539 | Polyline | DA | 21 | 22 | CONTINUOUS | 0.000000000( | 0.0000000 | | 44.9912718066 | 0105000000100000001 | 9 |
| 10 | 1540 | PolylineN | | | | | | | Estrada Nacional 123 | | 0105000000100000001 | 10 |
| 11 | 1541 | Polyline | | | | | | | Estrada Regional 2 | | 0105000000100000001 | 11 |
| 12 | 1542 | Polyline | | | | | | | Estrada N1 | | 0105000000100000001 | 12 |
| 13 | 1543 | Polyline | | | | | | | Estrada Municipal | | 0105000000100000001 | 18 |
| * | | | | | | | | | | | | |

**Figura 15 – Dados geográficos em *PostGIS*.**

- ***MySQL* –** O *MySQL*[24] implementa (a partir da versão 4.1) um subconjunto do ambiente *SQL with Geometry Types* proposto pelo OGC [MySqlMan06]. O termo *SQL with Geometry Types* refere-se a um ambiente SQL que tem sido estendido com um conjunto de tipos geométricos. Basicamente, foi adicionado um tipo novo (*geometry*) para as colunas da base de dados que guardam dados geométricos. A especificação descreve um conjunto de tipos geométricos SQL, bem como funções sobre esses tipos para criar e analisar valores geométricos. Desta maneira, nesse campo poderá guardar-se qualquer característica geográfica, como por exemplo, uma entidade (uma montanha, uma cidade,...), um espaço (uma área postal, os trópicos,...) ou uma localização específica (um ponto de interesse, a intercepção de duas estradas,...). Na Figura 17, apresentam-se os novos tipos introduzidos no *MySQL*, resultantes da adopção da norma SFS. Concretamente, pode-se usar a extensão SFS para definir uma tabela do género:

```
CREATE TABLE local (
  endereco CHAR(80) NOT NULL,
  localizacao POINT NOT NULL,
  PRIMARY KEY(endereco),
  SPATIAL KEY(localizacao)
);

Inserir nova entidade:

INSERT INTO local VALUES('Bom Jesus',
GeomFromText('POINT(2671 2500)'));
```

**Figura 16 – Criação de tabelas em *MySQL* para tipos geométricos.**

[24] http://www.mysql.org

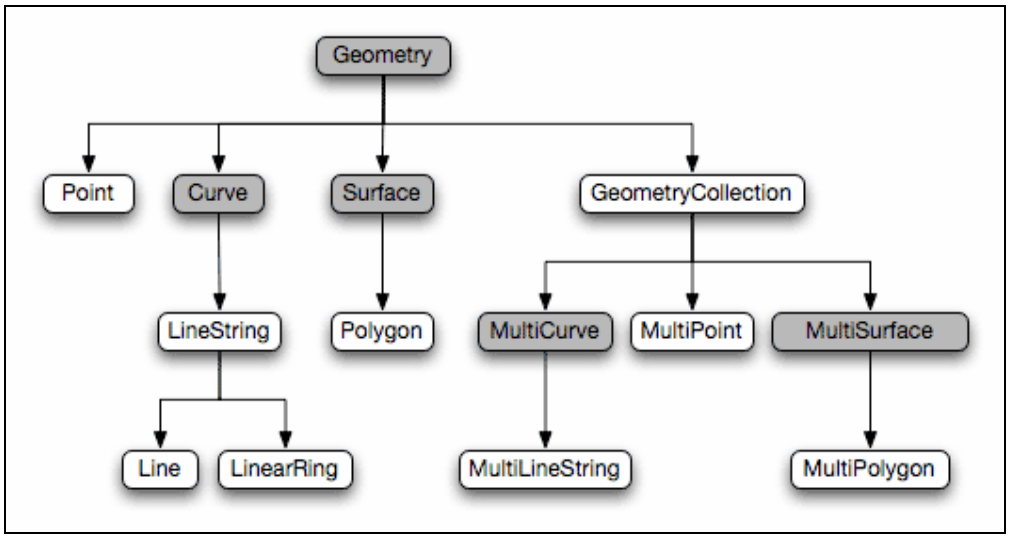



**Figura 17 – Novos tipos do *MySQL* para adopção da norma SFS.**

### 2.2.4. GML

O GML (*Geographic Markup Language*) é uma meta-linguagem codificada em XML, definida pelo OGC, para modelar a representação de entidades geográficas. O GML foi especificado simultaneamente para a modelação e distribuição de informação geográfica, incluindo quer as propriedades espaciais quer as não espaciais [OGC99].

O objectivo do GML é oferecer um conjunto de regras com as quais um qualquer utilizador de SIG pode definir sua própria linguagem para descrever os seus dados. Por isso a designamos de meta-linguagem. Para tanto, em GML começa-se por definir os esquemas XML (XML *Schema*) para o domínio concreto de aplicação. O esquema XML define os elementos, os atributos e a estrutura usados num documento que descreve os dados de um determinado domínio de aplicação. A versão GML 3.0 inclui esquemas que contêm os modelos de geometria, entidades e superfícies, que facilitam a definição dos esquemas concretos necessários em cada domínio de aplicação. Os esquemas padrão disponíveis, estão publicados nas especificações do OGC [Coxetal03] e os principais são os seguintes:

- ***BasicTypes*:** que engloba uma série de componentes simples e genéricos para representação arbitrária de atributos, nulos ou não;
- ***Topology*:** o qual especifica as definições do esquema geométrico dos dados, bem como sua descrição;
- ***CoordinateReference Systems*:** para sistemas de referência de coordenadas;
- ***Temporal Information and Dynamic Feature*:** este esquema estende aos elementos características temporais dos dados geográficos e suas funções dinamicamente definidas;
- ***Definitions and Dictionaries*:** definições das condições de uso dentro de documentos com certas propriedades ou informações de referentes à propriedade padrão;
- ***Metadata*:** este esquema é utilizado para definir as propriedades dos pacotes de dados.

Com base nestes esquemas predefinidos, um utilizador pode definir o seu próprio esquema para o seu domínio de aplicação concreto. Mas há algumas exigências a seguir para obter conformidade:

- Deve assegurar-se que os tipos usados são subtipos dos correspondentes tipos do GML: *gml:AbstractFeatureType* ou *gml:AbstractFeatureCollectionType* para entidades e *gml:AbstractGeometryType* ou *gml:GeometryCollectionType* para a geometria;
- Um esquema de aplicação não pode mudar o nome, definição ou tipo de dado dos elementos obrigatórios do GML;
- Definições de tipos abstractos podem ser livremente estendidas ou restritas;
- Esquema de aplicação deve estar disponível a qualquer um que receba os dados estruturados por aquele esquema;
- Os esquemas relevantes devem especificar um "*namespace*" que não deve ser *http://www.opengis.net/gml*.

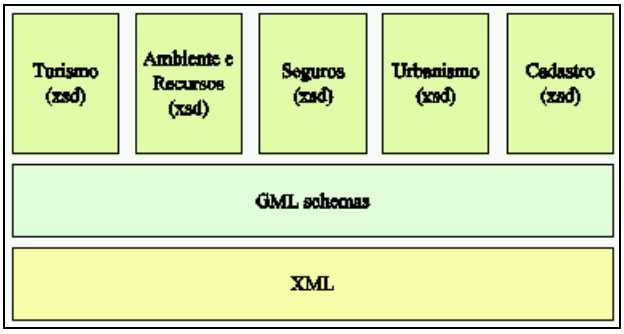



**Figura 18 - O GML como uma meta-linguagem em que se definem domínios de aplicação.**

Os esquemas do GML, por si só, não são adequados para criar uma instância de um documento. Estes devem ser estendidos pela criação de esquemas de aplicação para domínios específicos, seguindo as regras descritas na especificação. Esses esquemas denominam-se de perfis GML (GML *Profiles*) e que têm como objectivo simplificar a adopção do GML como formato standard. Foram publicados para uso público alguns perfis que seguem a especificação do GML, os mais importantes são:

- ***Point profile* –** para aplicações que usam só pontos como dados geométricos (sem necessidade de usar a gramática completa do GML);
- **GML *Simple Features profile* –** para aplicações que suportem entidades vectoriais e transacções via WFS (perfil mais completo que o anterior).

O GML *Simple Features profile* vem de raiz com muitas aplicações SIG, que suportam GML, e suporta vários objectos vectoriais tais como: pontos, linhas, polígonos, colecções geométricas (*MultiPoint* e *MultiPolygon*) definidos por coordenadas cartesianas uni, bi ou tridimensionais associados a eventuais sistemas de referência espacial e um modelo simplificado de entidades. No entanto, não disponibiliza suporte para: topologias, coberturas (*coverages*) e entidades dinâmicas, entre outras. Ainda assim, é o suficiente para cobrir grande parte dos problemas estudados nos SIG.

Uma vantagem no uso de XML é a flexibilidade oferecida para criar etiquetas que expressam o significado do dado descrito, obtendo-se um documento rico

semanticamente. Mas considere-se a seguinte situação: dois SIG de domínios diferentes representam uma determinada entidade, pelo GML, como <*rio*> e <*curso_de_agua*>. Na troca de dados entre os dois SIG, os esquemas também devem ser compartilhados, pois só assim uma aplicação poderá saber que <*rio*> ou <*curso_de_agua*> são da classe <*_Feature*> definida pelo esquema *Feature.xsd* do GML, e então processá-los adequadamente. Desta forma, o problema de acesso aos dados é resolvido. Mas não há como saber que <*rio*> é <*curso_de_agua*> e vice-versa. O aspecto semântico não é considerado de forma efectiva a promover a interoperacionalidade. Para amenizar este problema, pode acrescentar-se etiquetas que descrevem as entidades e as suas relações, ou que identifiquem sinónimos.

```
<abc:Building gml:id="SearsTower">
    <gml:name>Sears Tower</gml:name>
    <abc:height>52</abc:height>
    <abc:position>
        <gml:Point>
            <gml:coordinates>100,200</gml:coordinates>
        </gml:Point>
    </abc:position>
    <app:extent>
        <gml:Polygon>
            <gml:exterior>
                <gml:LinearRing>
                    <gml:coordinates>100,200</gml:coordinates>
                </gml:LinearRing>
            </gml:exterior>
        </gml:Polygon>
    </app:extent>
</abc:Building>
<abc:Building gml:id="SearsTower">
    <abc:position xlink:type="Simple" xlink:href="#p21"/>
</abc:Building>
<abc:SurveyMonument gml:id="g234">
    <abc:position>
        <gml:Point gml:id="p21">
            <gml:coordinates>100,200</gml:coordinates>
        </gml:Point>
    </abc:position>
</abc:SurveyMonument>
```

**Figura 19 - Exemplo de um documento GML.**

O GML já começa a ser amplamente utilizado, quer pela indústria quer pelos utilizadores ocasionais de IG. Isso deve-se ao facto de a sua especificação cobrir a grande maioria dos problemas do mundo real (o esforço do OGC, e todas a entidades envolvidas, tem sido enorme na promoção e desenvolvimento do GML) e ser, consequentemente, a melhor forma de representar IG. Para além disso, como partilha das características do formato XML é aceite por todas as aplicações que processam XML e é uma óptima forma de distribuir IG através *Web*, em particular para os WS [Rocha05]. Contrapondo com a distribuição, temos o armazenamento da IG que deve ser feito por motores muito eficientes (para que as consultas e alterações sejam rápidas), como os SGBD com suporte SFS previamente descritos.

Para uma visão mais detalhada à especificação do GML e suas características sugere-se consulta à dissertação de Jorge Rocha [Rocha05].

# 3. Publicação da IG na *Web*

No capítulo anterior mostraram-se as especificidades da IG, bem como alguns dos formatos mais conhecidos para armazenar dados geográficos (nomeadamente em formato vectorial).

Neste capítulo, propõe-se mostrar como publicar a IG na Internet. Começa-se por mostrar as vantagens que a *Web* trouxe à disseminação da IG. Começando pelo lado do servidor, resumem-se os conceitos orientadores do trabalho do OGC, para depois se focar e detalhar os principais serviços (Geo-WS) preconizados. De seguida, são apresentadas, muito sucintamente, duas aplicações *open source* estudadas para o efeito (*deegree* e *GeoServer*), implementações de referência para serviços especificados pelo OGC. Por fim, foca-se o lado do cliente (que utiliza um browser). Começa-se por apresentar dois clientes típicos. O primeiro desenvolvido pela equipa do *deegree*, o *iGeoPortal*, e o segundo, o *Mapbuilder*, desenvolvido em AJAX. Ainda na perspectiva do cliente, realça-se a adequação do SVG ao ambiente *Web* e à apresentação de IG.

A fechar o capítulo, apresenta-se a API do *Google Maps*, que representa um marco histórico na publicação de IG geográfica na *Web*. No capítulo seguinte, algumas destas tecnologias são aproveitadas num caso de estudo concreto, demonstrando a prontidão com que estas se podem incluir numa solução *Web*.

## *3.1. Introdução*

Os conteúdos dinâmicos e capacidades interactivas que caracterizam o actual sistema WWW trouxeram novas potencialidades para a publicação e exploração de IG, como sustentado em diversas publicações - [Çöltekinetal97], [PengBeimborn98], [Dogruetal04] e [Gomesetal06], entre outras. Nomeadamente, essas características expressam-se, resumidamente, em: maior acessibilidade; maior facilidade de actualização; novo meio de transmissão da informação; aumento das utilizações e redução de custos.

**Maior acessibilidade:** Em primeiro lugar, a Internet veio melhorar a acessibilidade à IG, com benefícios tanto para os produtores como para os utilizadores. Principalmente para instituições públicas produtoras de IG, a Internet veio oferecer um canal alternativo para a sua disseminação. As iniciativas de desenvolvimento de infra-estruturas nacionais de informação geográfica multiplicam-se um pouco por todo o mundo e demonstram o reconhecimento generalizado do papel da Internet na disseminação de IG.

**Maior facilidade de actualização:** Por se basear numa arquitectura cliente-servidor, a disponibilização de IG através da WWW pode beneficiar das vantagens de um controle centralizado dos dados. Estabelecendo um único ponto de manutenção, garante-se que a alteração e actualização dos dados fique imediatamente disponível para qualquer cliente WWW. A Internet veio possibilitar o acesso a informação em permanente actualização, em tempo (quase) real, de sensores remotos (e.g.

satélites meteorológicos) ou de câmaras de monitorização de actividades diversas (e.g. controle de tráfego automóvel);

**Meio de transmissão de informação:** Para além de melhorar a acessibilidade, a Internet pode também suportar o acesso distribuído à IG, isto é, nem sequer é necessário descarregar previamente a IG para o computador local. Esta possibilidade aplica-se tanto para informação gratuita, através da sua importação directa a partir dos servidores HTTP e FTP, como para informação com valor comercial, através da implementação de adequados mecanismos de comercialização.

**Redução de custos:** Do ponto de vista do produtor, a publicação e distribuição de IG através da Internet representa uma poupança significativa relativamente aos canais e processos tradicionais, uma vez que os custos de impressão e de distribuição são transferidos para o utilizador.

Os benefícios apresentados decorrem sobretudo da facilidade de acesso à informação e não da sua utilização prática. No entanto, é ainda possível expandir mais as potencialidades da Internet e da WWW, uma vez que para o utilizador, o acesso a IG apenas se torna verdadeiramente útil se dispuser de ferramentas que permitam a sua manipulação e exploração.

## *3.2. Conceitos*

O trabalho do OGC é enquadrado por um documento de referência, designado por OGC *Reference Model* [OGCRM03]. Este modelo descreve a orientação do trabalho do OGC no que diz respeito às especificações, implementação de soluções e avaliação da interoperacionalidade para serviços, dados e aplicações geoespaciais. Concretamente, o modelo tem como objectivos:

- Fornecer os fundamentos para a coordenação e compreensão (interna e externa ao OGC) das actividades correntes do OGC, bem como as bases técnicas;
- Actualização do guia do OpenGIS de 1998;
- Descrever as bases dos requisitos do OGC para a interoperacionalidade geoespacial;
- Descrever a arquitectura da *framework* do OGC através de séries de pontos de vista não sobrepostos: incluindo quer elementos já existentes quer elementos futuros;
- Regularizar o desenvolvimento de arquitecturas do domínio específico da interoperacionalidade através da disponibilização de exemplos.

As especificações do OGC baseiam-se, então, numa *framework* arquitectural chamada de OpenGIS *Services Framework* [Percivall03], que especifica o âmbito, objectivos e comportamento de uma série de componentes. Neste sentido, a *framework* representa uma arquitectura de referência para desenvolvimento de aplicações geográficas, no espírito da ISO 19119.

A definição dos componentes segue o paradigma de WS (*Web Services*) e, portanto, está sujeita a regras de concepção e de implementação. As regras de concepção incluem orientação a serviços, auto-descrição dos serviços e operações sem estados persistentes (*stateless operation*). As regras de implementação endereçam questões relativas a interoperacionalidade, incluindo a adopção de formatos de intercâmbio em XML e utilização de protocolos comuns à Internet.

A tecnologia WS propõe a exposição das transacções e das regras de negócios por meio de protocolos que podem ser acedidos e entendidos por qualquer linguagem de programação, em qualquer sistema operativo, sobre qualquer dispositivo [Costa02].

Na tecnologia dos WS, a disponibilização e acesso aos serviços envolvem basicamente três elementos: consumidores de serviços (clientes), produtores de serviços (fornecedores) e catálogo de serviços. Neste processo, cabe ao cliente enviar uma mensagem de maneira a descobrir o WS. Assim que descoberto, o cliente inquere o catálogo[25] que retorna uma descrição das funcionalidades do serviço. Essa informação permite ao cliente invocar o serviço, comunicando com o fornecedor, e reconhecer as mensagens aceites por este.

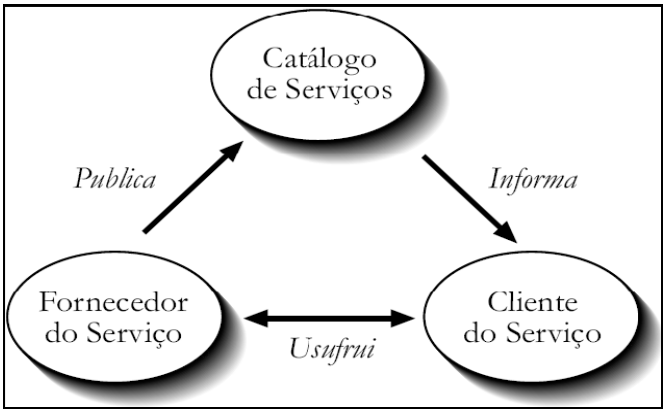



**Figura 20 – Arquitectura dos WS.**

A arquitectura apresentada necessita de normas que suportem a comunicação entre os três elementos mencionados. Nesse sentido, o consórcio W3C definiu algumas de grande importância e que Araújo [Araújo05] descreve em detalhe:

- XML – como linguagem para os documentos e mensagens trocadas;
- SOAP – como protocolo de comunicação;
- WSDL – para descrição dos serviços;
- UDDI – como serviço de registo e descoberta de WS.

### 3.2.1 *Geo Web Services*

Os serviços originalmente especificados pelo OGC, mais conhecidos por Geo-WS, não seguem as recomendações do W3C para definição de XML WS dado que o

[25] Nem todos os WS são reconhecidos pelo catálogo. Cada WS é que toma a decisão de ser ou não conhecido.

esforço para a normalização de WS é posterior aos Geo-WS [Araújo05]. Os Geo-WS não suportam SOAP para intercâmbio de dados (as mensagens são trocadas no protocolo HTTP através das operações GET e POST, usadas em qualquer navegador de Internet comum), WSDL para descrição dos serviços (usa-se uma operação comum aos Geo-WS para a descrição dos serviços, *GetCapabilities*) e UDDI para registro dos serviços (não existe noção de catálogo).

Mais recentemente, a série de propostas de especificação conhecidas colectivamente como OpenGIS *Web Service 2 initiative* definem interfaces que utilizam os padrões do W3C [Rocha05]. Porém, tais especificações ainda são tratadas como propostas de mudança, assistindo-se às primeiras tentativas de conformidade.

Os Geo-WS preconizados pelo OGC pretendem criar interfaces uniformizadas sobre toda a panóplia de serviços, aplicações e dados, para que os mesmos possam ser utilizados num ambiente *Web* [Rocha05].

Seguidamente serão apresentados dois serviços (os mais relevantes para o contexto desta dissertação) de entre os vários propostos pelo OGC: um serviço de entidades geográficas (WFS) e um serviço de mapas (WMS).

#### 3.2.1.1. WFS

A especificação OpenGIS *Web Feature Service* (WFS) [OGCWFS02] define um serviço para que clientes possam recuperar objectos (entidades) espaciais em formato GML de servidores WFS. O serviço pode ser implementado pelo servidor em duas versões: básica, onde apenas operações de consulta ficam disponíveis, ou transaccional, que implementa o serviço completo, que inclui operações de inserção, remoção, actualização, a par da consulta de objectos (entidades) geográficos.

A especificação define as operações e os requisitos necessários para a concepção do servidor WFS, em que o objectivo é estabelecer interoperacionalidade entre sistemas. A especificação WFS define:

- A utilização de HTTP (*Hypertext Transfer Protocol*) como meio de comunicação entre cliente e servidor;
- A utilização de documentos XML como interface para a troca de informações entre cliente e servidor;
- A utilização de GML para representação das entidades geográficas.

As seguintes operações são definidas para o serviço:

- ***getCapabilities*** - descreve as características do servidor (auto-descrição do serviço);
- ***describeFeatureType*** - descreve os tipos que caracterizam as entidades que podem ser servidas;
- ***getFeature*** - retorna as instâncias dos objectos disponíveis. O cliente pode seleccionar quais os objectos desejados por critérios espaciais ou baseados nas suas características.
- ***Transaction*** - utilizado para a execução de operações de modificação dos objectos (inserção, remoção e actualização).

- ***lockFeature*** **-** bloqueia uma ou mais instâncias durante uma transacção.

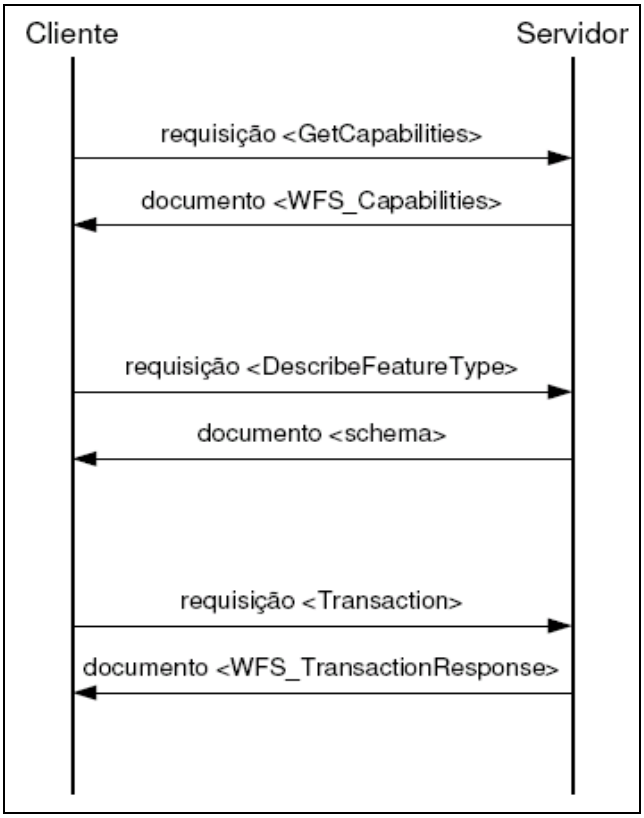



**Figura 21 - Exemplo de execução do serviço WFS.**

Todos os pedidos, e respectivas respostas, para as operações suportadas pelo serviço seguem os respectivos *XML Schemas* que podem ser encontrados no repositório do OpenGIS[26]. Araújo, na sua dissertação [Araújo05], trata exaustivamente esta questão da validação dos pedidos/respostas ao serviço através de XML *Schemas*. Seguidamente, apresentam-se alguns pedidos mais comuns ao serviço, pedidos esses que podem ser efectuados pelos protocolos GET ou POST do HTTP. Normalmente, para os pedidos mais simples é usado o protocolo GET (através de um *url* no *browser*). Os pedidos mais extensos e complexos, normalmente, são gerados em XML e enviados pelo protocolo POST ao servidor.

Então, para retornar a descrição do serviço pode-se fazer:

```
http://localhost:8888/GeoServer/wfs?Request=GetCapabilities
```

**Figura 22 – Pedido *GetCapabilities* a um WFS via GET.**

A resposta do serviço é um documento XML do tipo:

[26] http://schemas.opengis.net/wfs/1.0.0/


**Figura 23 – Resposta a um pedido *GetCapabilities* (1ª parte).**

Este documento contempla uma meta-descrição do serviço (<*Service*>), as operações disponíveis (<*Capability*>), a lista de entidades disponíveis (<*FeatureTypeList*>) e os filtros disponíveis (<*ogc:Filter_Capabilities*>). Dentro das entidades disponíveis tem-se algo do tipo:

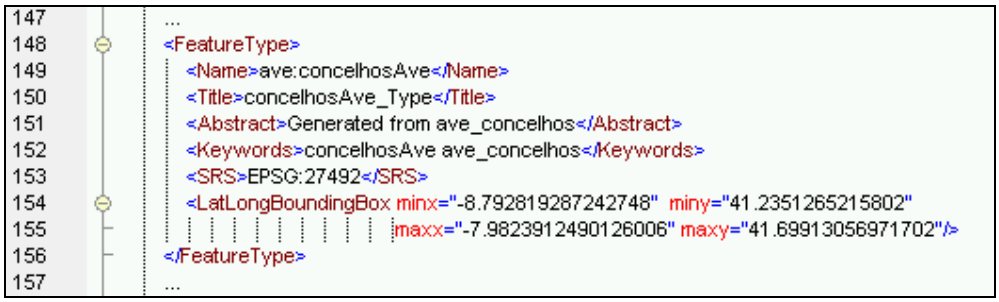



**Figura 24 – Resposta a um pedido *GetCapabilities* (2ª parte).**

Com base nas entidades disponíveis, pode-se gerar um pedido para dar informação sobre as instâncias destas, segundo determinados critérios:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:GetFeature service="WFS" version="1.0.0"
  outputFormat="GML2"
  xmlns:topp="http://www.openplans.org/topp"
  xmlns:wfs="http://www.opengis.net/wfs"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wfs
                      http://schemas.opengis.net/wfs/1.0.0/WFS-basic.xsd">
  <wfs:Query typeName="ave:concelhosAve">
    <ogc:Filter>
      <ogc:FeatureId fid="concelhosAve.3"/>
    </ogc:Filter>
    </wfs:Query>
</wfs:GetFeature>
```

**Figura 25 – Pedido *GetFeature* a um WFS via POST.**

Este pedido foi gerado em XML e enviado por POST ao servidor pedindo informação sobre o concelho com "*id*" igual a 3 definido na entidade "*ave:concelhosAve*". O resultado é o seguinte:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:FeatureCollection xmlns:wfs="http://www.opengis.net/wfs" xmlns:gml="http://www.opengis.net/gml"
    xmlns:ave="http://localhost/ave" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wfs
      http://localhost:8888/geoserver/schemas/wfs/1.0.0/WFS-basic.xsd
      http://localhost/ave
      http://localhost:8888/geoserver/wfs/DescribeFeatureType?typeName=ave:concelhosAve">
  <gml:boundedBy>
    <gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#27492">
      <gml:coordinates xmlns:gml="http://www.opengis.net/gml" decimal="." cs="," ts=" ">
        -41976.17049906,176657.26043185 -31223.19973704,186597.07996111</gml:coordinates>
    </gml:Box>
  </gml:boundedBy>
  <gml:featureMember>
    <ave:concelhosAve fid="concelhosAve.3">
      <ave:the_geom>
        <gml:MultiPolygon srsName="http://www.opengis.net/gml/srs/epsg.xml#27492">
          <gml:polygonMember>
            <gml:Polygon>
              <gml:outerBoundaryIs>
                <gml:LinearRing>
              </gml:outerBoundaryIs>
            </gml:Polygon>
          </gml:polygonMember>
        </gml:MultiPolygon>
      </ave:the_geom>
      <ave:OBJECTID>3</ave:OBJECTID>
      <ave:ID>51.0</ave:ID>
      <ave:concelho>Trofa</ave:concelho>
      <ave:distrito>PORTO</ave:distrito>
      <ave:area_ha>7187.89453536</ave:area_ha>
      <ave:n_freg>8</ave:n_freg>
      <ave:alt_max>254.0</ave:alt_max>
      <ave:alt_min>19.0</ave:alt_min>
      <ave:Shape_Leng>40802.6727371</ave:Shape_Leng>
      <ave:Shape_Area>7.18789450699E7</ave:Shape_Area>
    </ave:concelhosAve>
  </gml:featureMember>
</wfs:FeatureCollection>
```

**Figura 26 – Resposta a um pedido *GetFeature*.**

Este é um documento GML e contém toda a informação existente na estrutura de dados que guarda a IG, neste caso uma *shapefile*.

Um exemplo de um pedido transaccional, para inserir um novo registo pode ser do tipo (a geometria a ser inserida é um ponto):

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!--- Insere em Shapefile -->
3  <wfs:Transaction service="WFS" version="1.0.0"
4    xmlns:wfs="http://www.opengis.net/wfs"
5    xmlns:topp="http://www.openplans.org/topp"
6    xmlns:gml="http://www.opengis.net/gml"
7    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
8    xsi:schemaLocation="http://www.openplans.org/topp
9      http://localhost:8888/geoserver/wfs/DescribeFeatureType?typeName=topp:coord"
10 >
11   <wfs:Insert>
12     <topp:trofarecs>
13       <topp:the_geom>
14         <gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#27354">
15           <gml:coordinates decimal="." cs="," ts=" ">-42030,182030</gml:coordinates>
16         </gml:Point>
17       </topp:the_geom>
18       <topp:DESIG>Cruzeiro da Tocha</topp:DESIG>
19       <topp:icon>95.png</topp:icon>
20       <topp:NUM>232323</topp:NUM>
21     </topp:trofarecs>
22   </wfs:Insert>
23 </wfs:Transaction>
```

**Figura 27 – Pedido transaccional (inserção) a um WFS.**

A resposta do servidor é dada sob a forma de erro ou sucesso, tal como:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <wfs:WFS_TransactionResponse version="1.0.0" xmlns:wfs="http://www.opengis.net/wfs"
3      xmlns:ogc="http://www.opengis.net/ogc"
4      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5      xsi:schemaLocation="http://www.opengis.net/wfs
6        http://localhost:8888/geoserver/schemas/wfs/1.0.0/WFS-transaction.xsd">
7    <wfs:InsertResult handle="trofarecs 1">
8      <ogc:FeatureId fid="new0"/>
9    </wfs:InsertResult>
10   <wfs:TransactionResult>
11     <wfs:Status>
12       <wfs:SUCCESS/>
13     </wfs:Status>
14   </wfs:TransactionResult>
15 </wfs:WFS_TransactionResponse>
```

**Figura 28 – Resposta a um pedido transaccional.**

### 3.2.1.2. WMS

A especificação OpenGIS *Web Map Service* (WMS) define um serviço para a produção de mapas na Internet. Neste sentido, o mapa é uma representação visual dos dados geográficos. Os mapas produzidos são representações geradas em formatos de imagem, como PNG, GIF e JPEG, ou em formatos vectoriais, como o SVG.

Quando o cliente requisita um mapa utilizando o serviço, um conjunto de parâmetros deve ser passado ao servidor: as camadas desejadas, os estilos que devem ser aplicados sobre as camadas, a área de cobertura do mapa, a projecção ou sistema de coordenadas geográficas, o formato da imagem gerada e também o seu tamanho.

Um servidor WMS é um servidor que produz mapas de dados georeferenciados a partir de um conjunto bem definido de operações contido no documento da especificação [OGCWMS01].

A especificação WMS determina que um mapa é formado por um número de camadas[27] e estilos respectivos agrupados numa ordem específica, muito ao jeito da representação do mundo real adoptada pela maioria dos SIG. Uma camada pode ser considerada uma folha transparente que contém entidades representadas através de símbolos, onde a camada define as entidades e o estilo define como as entidades são representadas simbolicamente. Através de inclusão ou remoção dessas camadas, é possível obter mapas mais complexos ou mais simples. O WMS descreve a aparência de um mapa em termos dessas camadas.

A especificação WMS oferece um número finito de estilos predefinidos nos quais exibe as camadas de IG. A linguagem SLD (ver secção 3.2.2) é utilizada para permitir que clientes WMS possam definir as suas próprias regras de estilos, além de apenas escolher entre os estilos existentes na especificação do WMS.

Existem três operações definidas pela especificação WMS:

- ***GetCapabilities*:** através desta operação os clientes podem obter informações sobre o serviço oferecido pelo WMS, auxiliando a formulação de requisitos válidos e promovendo a independência de cada servidor;
- ***GetMap*:** possibilita recuperar a imagem de um mapa cujos parâmetros geoespaciais e dimensionais são bem definidos. Através do modelo proposto pela especificação é possível que um cliente solicite camadas individuais de mapas de diferentes servidores, viabilizando a criação de uma rede de servidores de mapas distribuídos;
- ***GetFeatureInfo*:** esta operação é opcional segundo a especificação. Pode ser utilizada para obter informações sobre determinadas entidades que são exibidas no mapa. Através dessas operações a especificação pretende padronizar a forma na qual mapas são requisitados por clientes e a forma na qual os servidores descrevem os dados por eles manipulados.

[27] Do inglês layer.

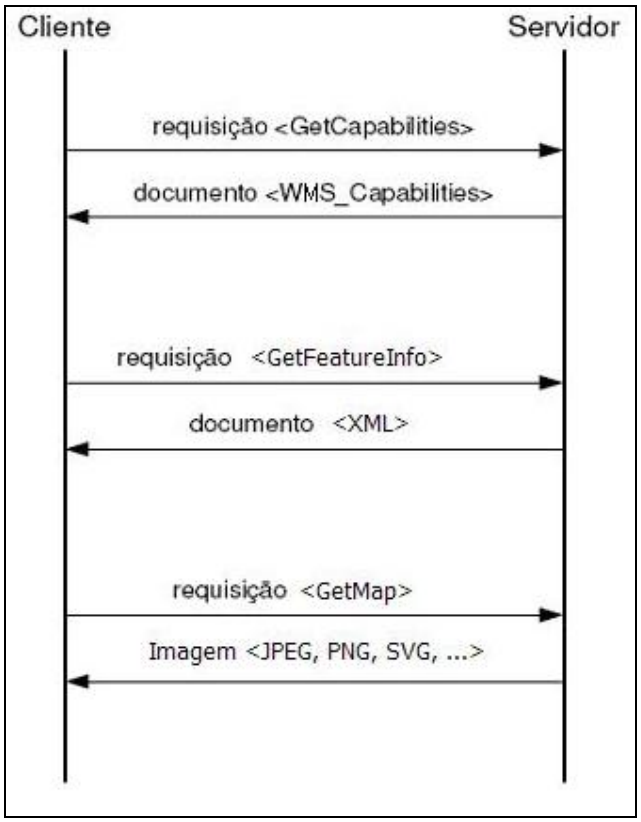



**Figura 29 - Exemplo de execução do serviço WMS.**

Seguidamente, apresentam-se alguns pedidos mais comuns ao serviço, pedidos que tal como para os pedidos ao WFS, podem ser efectuados pelos protocolos GET ou POST do HTTP.

Um pedido para retornar a descrição (formal) do próprio serviço, para se saber que funcionalidades e dados que o mesmo oferece, utiliza-se a operação *GetCapabilities* referida. Usando o protocolo GET, o pedido seria tipo:

```
http://localhost:8888/GeoServer/wms?request=GetCapabilities&servic
e=WMS
```

**Figura 30 – Pedido *GetCapabilities* a um WMS via GET**

A resposta do serviço é um documento XML que obedece a um formato específico disponível no repositório do OpenGIS[28] (Araújo, em [Araújo05], descreve ao pormenor todos os pedidos e respostas do serviço). Um extracto desse documento é aqui apresentado (a parte referente a uma camada disponibilizada).

[28] http://schemas.opengis.net/wms/1.1.1/

```
2894  <Layer queryable="1">
2895    <Name>ave:concelhosAve</Name>
2896    <Title>concelhosAve_Type</Title>
2897    <Abstract>Generated from ave_concelhos</Abstract>
2898    <KeywordList>
2899      <Keyword>concelhosAve ave_concelhos</Keyword>
2900    </KeywordList>
2901    <SRS>EPSG:4326</SRS>
2902    <!--WKT definition of this CRS:
2923    <LatLonBoundingBox minx="-8.792819287242748" miny="41.2351265215802"
2924      maxx="-7.9823912490126006" maxy="41.69913056971702"/>
2925    <Style>
2926      <Name>concelhos</Name>
2927      <Title>Default Styler</Title>
2928      <Abstract>A sample style that uses a filter, printing only the
2929        lines with a LENGTH property of over 5000.  This will work
2930        with the default bc_roads layer</Abstract>
2931      <LegendURL width="20" height="20">
2932        <Format>image/png</Format>
2933        <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
2934          xlink:href="http://localhost:8888/geoserver/wms/GetLegendGraphic?VERSION=1.0.0
2935                &amp;FORMAT=image/png&amp;WIDTH=20&amp;HEIGHT=20
2936                &amp;LAYER=ave:concelhosAve"/>
2937      </LegendURL>
2938    </Style>
2939  </Layer>
```

**Figura 31 – Extracto de uma resposta ao pedido G*etCapabilities*.**

Com base no extracto anterior, pode formular-se agora um pedido para mostrar a área definida por essa camada. Usa-se novamente o protocolo GET para formular o pedido.

```
http://localhost:8888/GeoServer/wms?service=WMS&VERSION=1.1.1&REQU
EST=GetMap&STYLES=concelhos&SRS=EPSG:4326&BBOX=-
8.792819287242748,41.2351265215802,-
7.9823912490126006,41.69913056971702&WIDTH=250&HEIGHT=250&FORMAT=i
mage/jpeg&EXCEPTIONS=application/vnd.ogc.se_inimage&LAYERS=ave:con
celhosAve
```

**Figura 32 – Pedido *GetMap* a um WFS via GET**

Deste pedido resultaria então um mapa com os limites dos concelhos da Região do Vale do Ave.

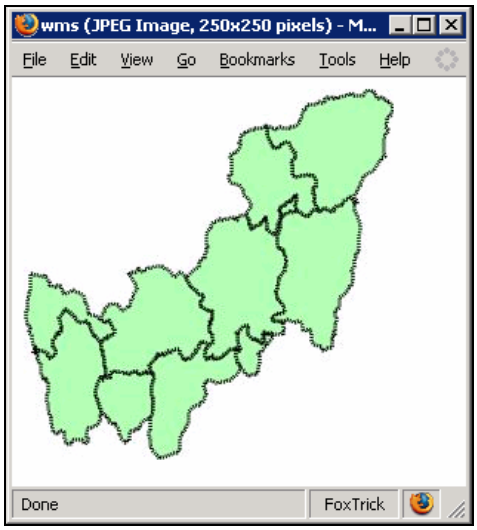



**Figura 33 – Resposta a um pedido *GetMap*.**

Para a sobreposição de várias camadas disponíveis basta acrescentar no pedido os estilos para cada camada em "*STYLES*" e o nome da camada em "*LAYER*" (a separação é feita por "," e é preciso ter em conta que o primeiro estilo definido corresponde à primeira camada definida, o segundo estilo corresponde à segunda camada e assim sucessivamente), tal como:

```
http://localhost:8888/GeoServer/wms?service=WMS&VERSION=1.1.1&REQU
EST=GetMap&STYLES=concelhos,rios&SRS=EPSG:4326&BBOX=-
8.792819287242748,41.2351265215802,-
7.9823912490126006,41.69913056971702&WIDTH=250&HEIGHT=250&FORMAT=i
mage/jpeg&EXCEPTIONS=application/vnd.ogc.se_inimage&LAYERS=ave:con
celhosAve,ave:rio_ave
```

**Figura 34 – Pedido *GetMap* com sobreposição de camadas.**

A resposta a este pedido seria:

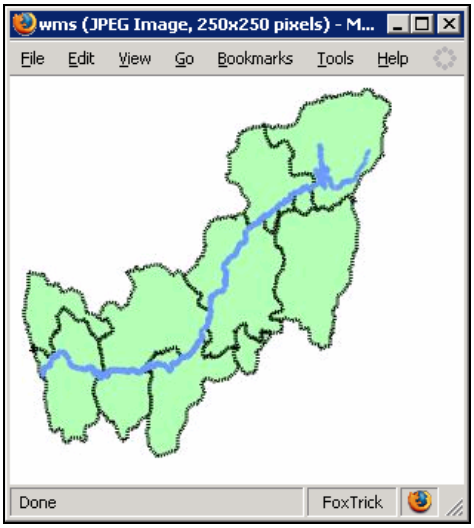



**Figura 35 – Resposta a um pedido *GetMap* (sobreposição de camadas).**

### 3.2.2. Apresentação da IG (SLD)

Já vem da cartografia a necessidade de criar representações de fácil percepção. É deveras importante mostrar ao utilizador mapas em que, por exemplo, a diferença entre estradas e rios, auto-estradas e caminhos-de-ferro, edifícios e zonas verdes, etc. esteja bem vincada ao nível visual (usando cores, formas, símbolos diferentes, etc.).

Nesta secção, define-se a forma como vão ser desenhados (mostrados) os dados geográficos. Isso é feito através de um documento SLD (*Styled Layer Descriptor*) [OGCSLD02] que é usado como extensão à especificação WMS (1.0 ou 1.1) para permitir a definição da simbologia e personalização dos dados geográficos. Estes são documentos XML.

A linguagem SLD é definida pelo OGC para representação gráfica de objectos espaciais (linhas, pontos, polígonos, etc.). Nesta linguagem é possível definir regras agrupando objectos em diferentes categorias e definindo para cada grupo um estilo diferente. Estes estilos podem ser utilizados por ferramentas de visualização que geram a simbologia associada a cada entidade geográfica. A especificação SLD é diferente da especificação GML. A especificação GML não se preocupa com a representação de uma entidade ou com a simbologia que essa entidade deve possuir. O GML restringe-se à descrição das entidades (conteúdos).

Para melhor compreensão dos objectivos e capacidades desta linguagem, tome-se em conta os seguintes exemplos:

1) Apresentação de pontos (usados para representar cidades ou pontos de interesse):

```
<?xml version="1.0" encoding="UTF-8"?>
<StyledLayerDescriptor version="1.0.0">
  <NamedLayer>
    <Name>Default Styler</Name>
    <UserStyle>
      <FeatureTypeStyle>
        <FeatureTypeName>Feature</FeatureTypeName>
        <Rule>
          ...
          <PointSymbolizer>
            <Graphic>
              <ExternalGraphic>
                <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                  xlink:type="simple" xlink:href="city2.png"/>
                <Format>image/png</Format>
              </ExternalGraphic>
              <Mark/>
              <Size>35.0</Size>
            </Graphic>
          </PointSymbolizer>
        </Rule>
      </FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

**Figura 36 – Pontos.sld**

```
<?xml version="1.0" encoding="UTF-8"?>
<StyledLayerDescriptor version="1.0.0">
  ...
        <PointSymbolizer>
          <Graphic>
            <Mark>
              <WellKnownName>circle</WellKnownName>
              <Fill>
                <CssParameter name="fill">#FFFFFF</CssParameter>
              </Fill>
              <Stroke>
                <CssParameter name="stroke">#000000</CssParameter>
                <CssParameter name="stroke-width">2</CssParameter>
              </Stroke>
            </Mark>
            <Opacity>
              <ogc:Literal>1.0</ogc:Literal>
            </Opacity>
            <Size>
              <ogc:Literal>6</ogc:Literal>
            </Size>
          </Graphic>
        </PointSymbolizer>
...
</StyledLayerDescriptor>
```

**Figura 37 – Pontos1.sld**

Na figura 34 pode-se ver como se define a apresentação de pontos através de uma imagem, enquanto que na figura 35 define-se a apresentação dos pontos através de um círculo. Os resultados destes dois ficheiros são:

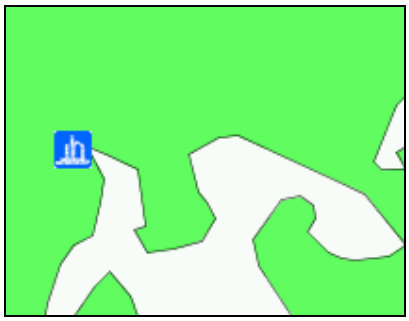


**Figura 38 – Visualização com Pontos.sld**

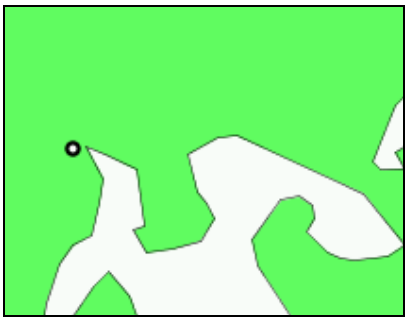


**Figura 39 – Visualização com Pontos1.sld**

2) Apresentação de polígonos (para representação de estradas, rios, limites de concelhos, etc.):

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <StyledLayerDescriptor version="1.0.0"
3      xsi:schemaLocation="http://www.opengis.net/sld StyledLayerDescriptor.xsd"
4      xmlns="http://www.opengis.net/sld" xmlns:ogc="http://www.opengis.net/ogc"
5      xmlns:xlink="http://www.w3.org/1999/xlink"
6      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
7    <NamedLayer>
8      <Name>TrofaRios</Name>
9      <UserStyle>
10       <Title>Rios</Title>
11       <Abstract>Estilo para mostrar rios</Abstract>
12       <FeatureTypeStyle>
13         <FeatureTypeName>Feature</FeatureTypeName>
14         <Rule>
15           <PolygonSymbolizer>
16             <Stroke>
17               <CssParameter name="stroke">#6E98FF</CssParameter>
18               <CssParameter name="stroke-width">3.0</CssParameter>
19             </Stroke>
20           </PolygonSymbolizer>
21         </Rule>
22       </FeatureTypeStyle>
23     </UserStyle>
24   </NamedLayer>
25 </StyledLayerDescriptor>
```

**Figura 40 – Estilo para desenho de rios (Rios.sld).**

Que resulta em:

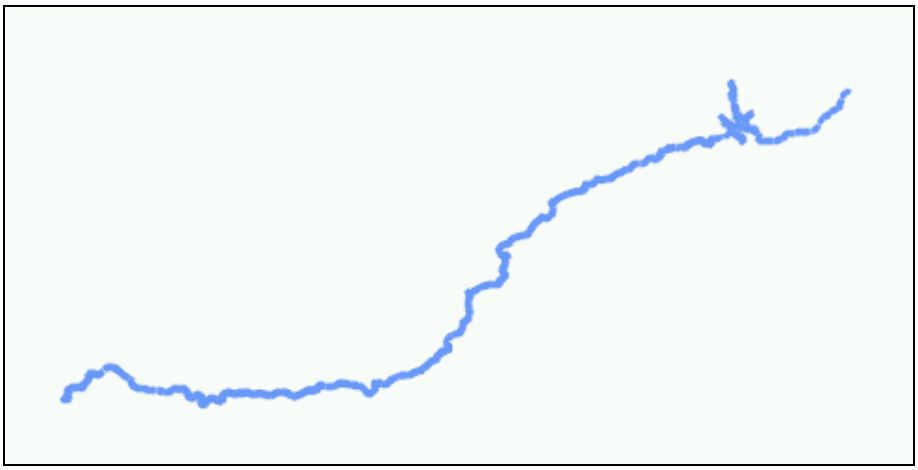


**Figura 41 – Visualização com Rios.sld**

3) Apresentação de texto (para representar nomes de países, cidades, etc.):

```
<?xml version="1.0" encoding="UTF-8"?>
<StyledLayerDescriptor xmlns="http://www.opengis.net/sld"
 xmlns:ogc="http://www.opengis.net/ogc" xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/sld
  http://schemas.cubewerx.com/schemas/sld/1.0.0/StyledLayerDescriptor.xsd" version="1.0.0">
  <NamedLayer>
    <Name>Default Styler</Name>
    <UserStyle>
      <Title>Default Styler</Title>
      <Abstract/>
      <FeatureTypeStyle>
        <FeatureTypeName>Feature</FeatureTypeName>
        <Rule>
          <TextSymbolizer>
            <Geometry>
              <ogc:PropertyName>the_geom</ogc:PropertyName>
            </Geometry>
            <Label>
              <ogc:PropertyName>DESIG</ogc:PropertyName>
            </Label>
            <Font>
              <CssParameter name="font-family">Arial</CssParameter>
              <CssParameter name="font-family">Sans-Serif</CssParameter>
              <CssParameter name="font-style">italic</CssParameter>
              <CssParameter name="font-size">10</CssParameter>
            </Font>
            <Fill>
              <CssParameter name="fill">#0000FF</CssParameter>
            </Fill>
          </TextSymbolizer>
        </Rule>
      </FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

**Figura 42 – Estilo para representação de texto (nomeconc.sld).**

A figura 40 mostra como se define um estilo para representação de texto. O texto a mostrar refere-se à geometria das entidades geográficas guardadas numa estrutura de dados (no caso deste exemplo, uma *shapefile*) e, nomeadamente, ao campo "*DESIG*". O resultado da aplicação deste estilo seria:


**Figura 43 – Visualização com nomeconc.sld**

4) Condições:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <StyledLayerDescriptor version="1.0.0"
3    xsi:schemaLocation="http://www.opengis.net/sld StyledLayerDescriptor.xsd"
4    xmlns="http://www.opengis.net/sld" xmlns:ogc="http://www.opengis.net/ogc"
5    xmlns:xlink="http://www.w3.org/1999/xlink"
6    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
7    <NamedLayer><Name>Default Styler</Name>
8    <UserStyle>
9      <Title>Tipos de recurso</Title>
10     <FeatureTypeStyle>
11       <FeatureTypeName>Feature</FeatureTypeName>
12       <Rule>
13         <ogc:Filter>
14           <ogc:PropertyIsEqualTo>
15             <ogc:PropertyName>Type</ogc:PropertyName>
16             <ogc:Literal>Castelo</ogc:Literal>
17           </ogc:PropertyIsEqualTo>
18         </ogc:Filter>
19         <PointSymbolizer>...</PointSymbolizer>
20       </Rule>
21       <Rule>
22         <ogc:Filter>
23           <ogc:PropertyIsEqualTo>
24             <ogc:PropertyName>Type</ogc:PropertyName>
25             <ogc:Literal>Igreja</ogc:Literal>
26           </ogc:PropertyIsEqualTo>
27         </ogc:Filter>
28         <PointSymbolizer>...</PointSymbolizer>
29       </Rule>
30       <Rule>
31         <ogc:ElseFilter/>
32         <PointSymbolizer>...</PointSymbolizer>
33       </Rule>
34     </FeatureTypeStyle>
35   </UserStyle>
36   </NamedLayer>
37 </StyledLayerDescriptor>
```

**Figura 44 – Exemplos de testes de condição para estilos (testes.sld).**

A figura 42 mostra um exemplo de como usar condições em SLD. Nomeadamente, testa-se se o campo "*Type*" (da estrutura de dados) contém as palavras "Castelo" ou "Igreja" e aplica-se um símbolo distinto para cada uma das ocorrências. Caso não tenha nenhuma das palavras aplica-se um símbolo por definição (linhas 31 e 32). O resultado seria algo do tipo:

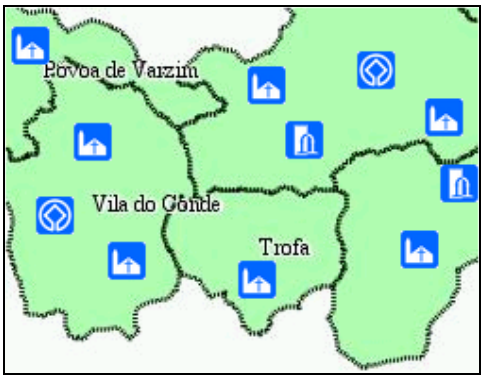



**Figura 45 – Visualização com testes.sld**

Obviamente que os exemplos acabados de apresentar são bastante simples, mas cobrem o essencial para a visualização dos objectos espaciais que definem a IG. Para uma visão detalhada da linguagem sugere-se consulta da especificação (que contém inúmeros exemplos).

A actual especificação WMS define três maneiras diferentes para os clientes de servidores de mapas poderem usar estilos:

- Os clientes especificam estilos já predefinidos no servidor através do pedido por HTTP GET (tipo http://.../wms?...&STYLES=nome_estilo&...);
- Os clientes definem os seus estilos pretendidos no próprio pedido por HTTP GET usando o parâmetro **SLD_BODY** (tipo http://.../wms?...&STYLES=<SLD_BODY>...</SLD_BODY>&...);
- Os clientes definem os seus estilos pretendidos gerando ficheiros SLD, incluem esses ficheiros num pedido XML *GetMap* e enviam o pedido por HTTP POST.

### 3.2.2. Salvaguarda de contextos de IG (WMC)

Considere-se a situação em que um utilizador está a trabalhar com um cliente genérico de mapas *Web*, no qual está a mostrar várias camadas provenientes de um ou mais servidores WMS. O utilizador também já tem escolhido uma região específica, uma escala e a ordem das camadas a serem mostradas. Suponha-se agora que o utilizador quer continuar o trabalho num dia posterior com os parâmetros de visualização que dispõe agora. Não é prático que este tenha de voltar a fazer as operações de controlo sobre os dados geográficos até ao estado em que estava. Para isso recorre-se a documentos WMC (*Web Map Context*), que são documentos XML que gravam o estado de visualização de mapas. Assim, o tal utilizador pode gravar o estado do seu trabalho de visualização e depois carregar esse estado num outro qualquer cliente WMS.

Um documento WMC define uma especificação do lado do cliente WMS (que é carregado tendo como base um conjunto de parâmetros de visualização predefinidos) e não é interpretado do lado do servidor. Deste modo, o cliente deve saber interpretar o documento WMC e transformá-lo em pedidos ao servidor WMS.

A especificação WMC [OGCWMC03] define como um conjunto de mapas provenientes de um ou mais servidores WMS pode ser descrito num formato independente de plataforma para armazenamento e transmissão. Para isto, é definido um documento XML denominado de contexto (*context*). Um documento de contexto inclui informações como: servidores que fornecem as camadas que compõe o mapa, área geográfica (*extent*) que deve ser apresentada, sistema de coordenadas de visualização, tamanho do mapa.

A especificação WMC tem como finalidades:

- Fornecer informações de inicialização de um mapa a uma determinada classe de clientes;
- Gravar o contexto corrente de um mapa visualizado em determinado cliente;
- Armazenar, além das informações de contexto corrente, informações adicionais sobre cada camada (estilos disponíveis, formatos e sistema de coordenadas), de forma a evitar consultas excessivas ao servidor de mapas;
- Recriar num cliente um contexto gerado num outro cliente diferente.

Os componentes de um documento WMC são:

- Área a visualizar (área do ecrã);
- Extensão geográfica da região a visualizar (limites geográficos);
- Informações sobre o autor do documento;
- Uma lista ordenada de camadas a mostrar, indicando o URL do servidor, nome da camada, indicações sobre se cada camada está visível ou pode ser interrogada e informações de estilo.

Dependendo de cada cliente WMS, o utilizador pode ou não ter a possibilidade de gravar ou ler documentos WMC. O verdadeiro sentido dos documentos WMC é a possibilidade de se armazenarem e catalogarem mapas predefinidos de interesse comum (a qualquer grupo de utilizadores dos vários contexto de IG) para serem pesquisados e disponibilizados.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<ViewContext version="1.0.0" id="atlas_world" xmlns="http://www.opengis.net/context" xmlns:xlink="http://www.w3.org/1999/xlink"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://www.opengis.net/context http://schemas.opengis.net/context/1.0.0/context.xsd">
  <General>
    <Window width="400" height="400"/>
    <BoundingBox SRS="EPSG:4326" minx="-8.63435745239258" miny="41.258731842041" maxx="-8.5062084197998" maxy="41.3448638916016"/>
    <Title>Trofa, Ave</Title>
    <KeywordList>
      <Keyword>Trofa</Keyword>
      <Keyword>Ave</Keyword>
    </KeywordList>
    <Abstract>Low resolution map of Trofa, Ave derived from the Digital Chart of the World.</Abstract>
  </General>
  <LayerList>
    <Layer queryable="0" hidden="0">
      <Server service="OGC:WMS" version="1.1.1" title="Layers">
        <OnlineResource xlink:type="simple" xlink:href="http://localhost:8888/geoserver/wms"/>
      </Server>
      <Name>topp:freguesias</Name>
      <Title>Trofa Freguesias</Title>
      <Abstract>Trofa Freguesias.</Abstract>
      <SRS>EPSG:4326</SRS>
      <FormatList>
        <Format current="1">image/png</Format>
      </FormatList>
      <StyleList>
        <Style current="1">
          <Name>concelhos</Name>
          <Title>freguesias</Title>
          <Abstract/>
        </Style>
      </StyleList>
    </Layer>
    <Layer queryable="1" hidden="0">
      <Server service="OGC:WMS" version="1.1.1" title="Layers">
        <OnlineResource xlink:type="simple" xlink:href="http://localhost:8888/geoserver/wms"/>
      </Server>
      <Name>topp:estradas</Name>
      <Title>Trofa Estradas</Title>
      <Abstract>Trofa Estradas.</Abstract>
      <SRS>EPSG:4326</SRS>
      <FormatList>
        <Format current="1">image/png</Format>
      </FormatList>
      <StyleList>
        <Style current="1">
          <Name>simple_roads</Name>
          <Title>estradas</Title>
          <Abstract/>
        </Style>
      </StyleList>
    </Layer>
  </LayerList>
</ViewContext>
```

**Figura 46 – Exemplo de um documento WMC (mapa de contexto da Trofa).**

## *3.3. Implementações open source* de servidores Geo-WS

Depois de apresentados os conceitos inerentes ao entendimento do que existe para a disponibilização de IG via *Web* (nomeadamente através das soluções preconizadas pelo OGC), foca-se agora o que existe para suportar os serviços descritos anteriormente. Foram estudadas duas soluções não proprietárias para servidor de Geo-WS, os projectos *deegree* e *GeoServer*, até se optar por um deles para sustentação do caso de estudo descrito nesta dissertação.

### 3.3.1. *Deegree*

O *deegree*[29] é a implementação de referência dos serviços de WMS 1.1.1 do OGC. Por ser uma aplicação Java, o *deegree* é independente de plataforma e pode ser utilizado em redes heterogéneas.

A estruturação do projecto *deegree* foi concebida como uma *framework* de desenvolvimento J2EE, integrado modularmente para a implementação de infra-estruturas de dados espaciais corporativas distribuídas.

Diferentemente de outros projectos de servidores de mapas para Internet, o *deegree* foi concebido desde o início para estar totalmente de acordo com as especificações do OGC, o que facilita a sua integração em ambientes corporativos heterogéneos.

Ainda, as interfaces para a leitura e escrita de dados permite a utilização de *Oracle Spatial*, *PostGIS*, *MySQL Spatial Extension*, *Shapefile* da ESRI, assim como bases de dados via JDBC, incluindo o acesso a ortofotos e dados *raster* nos mais diversos formatos.

O *deegree* oferece os seguintes serviços:

- **Serviços de Mapas (WMS)**: O Serviço WMS permite a criação de mapas em ambiente *Web* a partir de dados geográficos vectoriais e *raster*, de diferentes fontes, num único mapa, disponibilizado via rede para clientes *desktop*, navegadores *Web* e aplicações embebidas.
- **Serviços de Entidades (WFS)**: Serviço de dados geográficos vectoriais em formato GML 2.1.1, permitindo o seu processamento em aplicações clientes, como é o caso da edição vectorial em SIG.
- **Serviços de *Rasters* (WCS)**: Serviço que permite o acesso a dados geográficos *raster* (*coverages*), em diversos formatos, permitindo o seu processamento.
- **Serviços de Catálogo (WCAS)**: Permite a administração e consulta de meta-dados descritivos de dados e serviços geográficos, utilizando critérios textuais e espaciais.
- **Serviços de Procura (WFS-G)**: Este serviço permite a procura georeferenciada (*Gazetteer*) de feições utilizando identificadores textuais (nomes, etc.).
- **Serviços de Visualização de Terreno (WTS)**: Permite a criação de visualizações 3D de modelos de superfície (cidades, MDT, etc.) em navegadores *Web*.
- **Serviços *Web* de Transformação de Coordenadas (WCTS)**: Permite a transformação de coordenadas geográficas em ambiente *Web*.

[29] http://deegree.sourceforge.net

### 3.3.2. *GeoServer*

O projecto *GeoServer*[30] é também uma implementação em Java (J2EE) da especificação OGC WFS, que inclui transacções e que integra um WMS. É software livre e disponível sobre licença GPL 2.0. Trabalha com 4 padrões do OGC: SFS (*PostGIS*), WFS, WMS e GML e vem com suporte à manipulação de IG guardada sob as seguintes formas: Bases de Dados com extensões GIS (*Oracle*, *PostGIS* e *MySQL*), *Shapefiles* ou ArcSDE. O suporte a transacções é importante para disponibilizar a edição de *features* sobre uma rede de comunicações.

O *GeoServer* é a implementação de referência para WFS do OpenGIS e passa todos os testes CITE (*Conformance, Interoperability and Testing Environment*) do OpenGIS para WFS-T. É de simples instalação e, devido à sua interface gráfica para administração via *Web*, a configuração também não apresenta grandes problemas.

O *GeoServer* está estruturado segundo quatro directorias permanentes, duas das quais essenciais e mais importantes: ***data/*** - onde ficam guardados todos os dados relativos à IG e sua configuração; ***WEB-INF/*** - onde estão guardados todos os ficheiros de configuração do sistema, as classes de execução e todos os ficheiros necessários à ferramenta de administração via *Web*.

Embora o *deegree* disponibilize mais serviços (a maioria desses serviços não seriam usados para o caso de estudo descrito nesta dissertação), optou-se por se usar o *GeoServer* devido à facilidade de configuração e utilização e, acima de tudo, devido ao facto de suportar WFS-T que era exactamente o pretendido para o *site turismonoave.com*. Numa primeira fase, quando foi testado o *deegree,* este ainda não estava totalmente estável (versão 1.0) principalmente quando se tentava configurar este para trabalhar com as extensões GIS do *MySQL* (não se conseguia ler nem alterar a IG guardada) e mesmo depois de vários contactos com a equipa de desenvolvimento, não se consegui resolver totalmente essa questão. Nessa altura, a equipa de desenvolvimento do *deegree* prometeu o suporte total do *MySQL* para a versão seguinte do *deegree*. Curiosamente, quando se testou o *GeoServer*, também não produziu grande sucesso com as extensões GIS do *MySQL* (conseguia-se ler mas não alterar a IG guardada) levando a deixar de parte por completo essa solução e passou-se a guardar a IG em *shapefiles,* cujo suporte funcionava na perfeição (quer para leitura quer para alteração da IG).

## *3.4. A IG em SVG*

Hoje em dia, a maioria dos sistemas de visualização de mapas na *Web* (clientes), que permitem um certo tipo de interactividade, envolvem duas abordagens distintas: *Javascript*[31] (mais recentemente é usado também AJAX, como no caso do *Google Maps* ou do *Mapbuilder*) ou *scripts* (como a que foi criada para o caso de estudo desta dissertação) que vão fazendo pedidos em sequência aos servidores WMS e mostram as imagens geradas nos *browsers*. O uso de *Javascript* é excelente sob o ponto de vista da interactividade, mas pode sofrer de alguma falta de compatibilidade

[30] http://GeoServer.sourceforge.net
[31] Caso dos projectos *iGeoportal* e *MapBuilder* descritos no próximo capítulo.

entre *browsers* (casos de *Javascript* diferir de IE para *Firefox*). A segunda abordagem é pouco eficiente sob o ponto de vista da interactividade dado que, para cada operação de interacção com o mapa (*zoom*, *pan*, controlo de camadas, etc.), o cliente tem sempre de esperar pelo ciclo pedido/resposta do(s) servidor(es) WMS.

Uma alternativa viável a estas duas abordagens é a disponibilização, por parte dos servidores de IG, de mapas no formato SVG.

SVG (*Scalable Vector Graphics*) é um formato de ficheiros baseado em XML e é uma linguagem de marcação para descrever gráficos vectoriais bidimensionais [Bray00]. Foi criado pelo W3C e permite três tipos de objectos gráficos: formas vectoriais (círculos, rectângulos, polígonos, linhas e curvas), imagens e texto. Estes objectos podem ser agrupados, transformados e animados.

```
1  <svg xmlns="http://www.w3.org/2000/svg"
2       xmlns:xlink="http://www.w3.org/1999/xlink"
3       width="450" height="450">
4    <g>
5      <text x="200" y="30">Shape fill</text>
6      <polyline points="20,30 50,80 90,190 210,315"/>
7      <path d="M20,0 l20,0 l20,20 l0,20 l-20,20 l-20,0 l-20,-20 l0,-20 "/>
8    </g>
9  </svg>
```

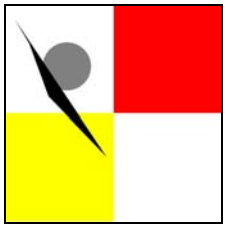


**Figura 47 – Ficheiro SVG e respectiva visualização.**

O formato vectorial SVG não é apresentado directamente na maioria dos *browsers*, necessita de *plugin*, e não foi criado com o propósito específico de visualizar IG. No entanto, é uma alternativa relevante para a publicação de dados geográficos na *Web*, principalmente por preservar a geometria das entidades geográficas possibilitando assim melhor interactividade e reaproveitamento. Além disto, é escrito segundo o padrão XML (portanto não é proprietário) e é portável (adequado à utilização na *Web*).

Inclui vários recursos de animação e apresentação de gráficos complexos, permite a inclusão de *scripts*, suporta a utilização de estilos (CSS) e possui sistemas de coordenadas e funções de transformação de sistemas de coordenadas. A possibilidade de incluir *scripts* num documento SVG aumenta a interactividade permitindo, entre outros, o controlo sobre os elementos do documento (essencial para mostrar/esconder diferentes características geográficas em mapas). Permite também o processamento das coordenadas vectoriais como simplificação de geometria para promover a transmissão dos dados [Faria02].

A utilização de SVG traz vantagens relativamente ao uso de imagens *raster* para mostrar IG, nomeadamente:

- **Formato de texto** – Por ser XML é muito mais compacto do que os formatos *raster*. Além disso, os ficheiros SVG podem ser indexados pelos motores de busca;
- **Escalável** – Como formato vectorial, as imagens em SVG estão sempre disponíveis em alta resolução em qualquer que seja a resolução e o nível de zoom, ao contrário das imagens *raster*;
- **Interacção e Animação** – Para além de operações de *zoom* e *pan* que são efectuadas de uma maneira rápida, sem degradar a imagem e sem consultar o

servidor, o formato SVG permite gráficos dinâmicos e interactivos. O utilizador pode remover, adicionar, alterar elementos dos mapas sem necessidade de novos pedidos ao servidor;

- **Texto seleccionável e pesquisável** – Ao contrário das imagens *raster* o texto em SVG pode ser seleccionado e alvo de pesquisa. Os utilizadores podem procurar por palavras-chave nos documentos SVG como, por exemplo, nomes de cidades ou ruas;
- **Formato standard e aberto** – O formato SVG é uma recomendação do W3C que coordena o esforço de várias empresas empenhadas no seu desenvolvimento e continuação como formato livre. Para além disso, como é baseado em XML, oferece todas as vantagens inerentes ao XML e às aplicações que processam XML.

Para atestar a popularidade e crescente utilização de SVG para visualizar IG, pode-se citar uma aplicação que foi desenvolvida no laboratório SIG da Universidade do Minho para visualizar mapas SVG em PDAs. A aplicação foi desenvolvida para a PT Inovação em 2002 (ainda muito no início da expansão do SVG) em JAVA e com suporte ao *profile SVG Basic*[32] (que não foi contemplado integralmente, mas cujas funcionalidades cobrem a quase totalidade das utilizações concretas). O projecto tinha como objectivo a visualização de informação geográfica em PDA e, em termos de funcionalidades, a aplicação permite as operações típicas de um visualizador de mapas, no caso: ampliar e reduzir, ampliar uma área, ampliar em percentagem, reverter para a ampliação anterior, arrastar, rodar, ajustar ao ecrã e procurar texto. Acrescentou-se ainda a possibilidade do utilizador alterar algumas preferências para acelerar a visualização em detrimento da qualidade dos elementos de texto.

O desenvolvimento do visualizador originou dois trabalhos de estágio final de curso: [Faria02] e [Rocha03]; duas publicações: [Rocha03] e [Martinsetal03]; e contribuiu para uma dissertação de doutoramento: [Rocha05]. Presentemente, e com o consentimento da PT Inovação, o projecto é livre sobre uma licença GPL e encontra-se disponível no *sourceforge*[33].

[32] http://www.w3.org/TR/SVGMobile/
[33] http://opensvgviewer.sourceforge.net/home.php

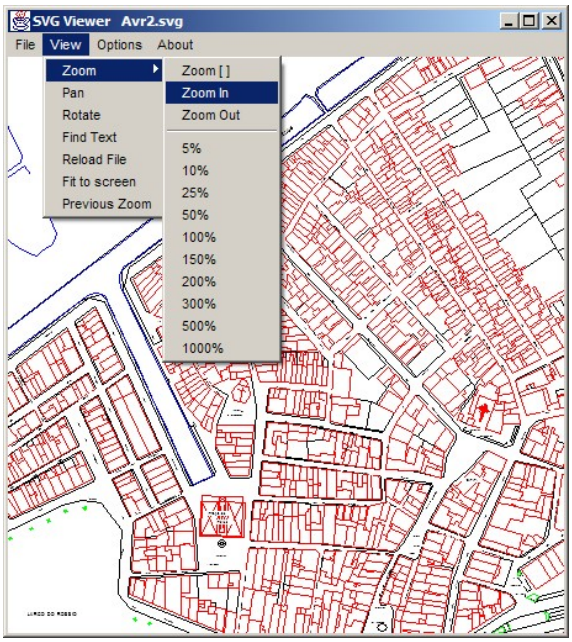



**Figura 48 – Visualizador de mapas SVG para PDAs (SVG Viewer).**

## *3.5. A API do Google Maps*

Um passo muito importante na disponibilização de IG, gratuita e à escala global, foi dado pelo *Google* quando lançou o *site Google Maps* e mais tarde com a aplicação *Google Earth*[34]. No entanto, o que leva a afirmar que se trata de um marco histórico para a disponibilização de IG gratuita, não é tanto o site em si, mas sim o facto de paralelamente o *Google* ter criado uma forma de os utilizadores de IG poderem incluir nos seus próprios sites, os mapas disponibilizados por este, sem a necessidade de nenhum componente extra no servidor *Web*.

Assim, a API do *Google Maps* é uma interface gratuita para utilizadores que pretendam usar mapas disponibilizados pelo *Google* nos seus sites através de *Javascript*. A API é muito potente e versátil, permitindo usar todas as opções que se encontram disponíveis no site do *Google Maps* (entre elas: *zoom*, *pan*, mapas de satélite, vectoriais e híbridos, marcadores de locais úteis, legendas, etc.). Para usar a API é preciso fazer um pedido de uma chave única para cada site onde se pretende usar. A documentação[35] detalhada da API contém explicações muito simples e muitos exemplos de como usar a API, sendo que na secção 4.3.4 desta dissertação, mostra-se um caso concreto de aplicação da API (versão 1) para visualizar IG.

[34] http://earth.google.com/
[35] http://www.google.com/apis/maps/documentation/

# 4. Caso de Estudo

No capítulo anterior, viu-se como a IG pode ser disponibilizada e publicada em ambiente *Web*, quer na componente do servidor quer nas tecnologias do lado do cliente. Nesse contexto, é apresentado agora um caso concreto de aplicação das teorias descritas anteriormente.

O site Turismo no Ave surgiu como um projecto encomendado pela ADRAVE[36], no âmbito da promoção do turismo na região do Vale do Ave, com o intuito de disponibilizar *online* os dados geográficos (para um conjunto de recursos já existentes numa BD *FileMaker*[37]). Para além disso, a BD deveria ser actualizada por todos os intervenientes (os responsáveis pelo turismo em cada uma das 10 câmaras municipais participantes) e não só pela própria ADRAVE. Este tipo de requisitos originou um estudo para se saber quais as melhores soluções SIG para esse efeito e que são agora apresentadas.

Neste capítulo descrevem-se as opções tomadas, com o objectivo de disponibilizar a IG na *Web*, juntamente com a restante informação sobre os recursos turísticos. No capítulo seguinte, retoma-se o mesmo caso de estudo para especificamente tratar a actualização da IG através da *Web*, num ambiente multi-utilizador, inspirado na metáfora *wiki*.

## *4.1. O site turismonoave.com*

O site *turismonoave.com* é um portal de turismo e a sua função é a promoção dos recursos turísticos da Região do Vale do Ave (composta por dez municípios). Com recursos turísticos pretende-se dizer todas as infra-estruturas e serviços que directa ou indirectamente têm a ver com turismo, como por exemplo: alojamento, gastronomia, património, recursos naturais, artesanato, serviços turísticos, etc.

No site, os utilizadores podem navegar por categorias de recursos turísticos e, na informação de cada recurso, entre outras, é disponibilizado um mapa de contexto desse recurso (sempre que os administradores do site queiram que esse recurso esteja visível). Para isso, foi montado um sistema SIG, que permite mostrar aos utilizadores a localização dos recursos turísticos, bem como toda a área envolvente de todos os concelhos abrangidos.

A necessidade de construir o SIG originou a que se tivesse desenvolvido um estudo sobre quais as tecnologias a usar para disponibilizar IG de uma maneira rápida e intuitiva aos utilizadores do site. Os diferentes tipos de utilizadores "obrigaram" à disponibilização de IG, mas também à necessidade de alterar essa informação sempre que se justificasse e de um modo simples e transparente (o objectivo era criar ferramentas para administração do site e da IG para pessoas com conhecimentos reduzidos em tecnologias *Web* e SIG).

[36] http://www.adrave.pt/
[37] http://www.filemaker.com/

O site foi construído sobre a plataforma *TikiWiki*[38] (PHP + *MySql*). Na BD são guardados os recursos numa tabela e, entre outra informação, são guardadas as coordenadas projectadas (Sistema *Hayford-Gauss Datum* 73) de cada recurso para posterior tratamento.

Naturalmente que os Geo-WS se revelaram como a melhor opção para o suporte à IG no site. Do ponto de vista pessoal, a participação da Universidade tem que promover a utilização das tecnologias mais actuais, pelo que pareceu óbvio a utilização de serviços OGC, para integrar mapas no meio do resto da informação disponibilizada pelo site, com algumas vantagens. A primeira tem a ver com a abordagem típica modular, em que se pode separar o problema da actualização e disponibilização da IG do restante trabalho mais relacionado com o desenvolvimento de um site dinâmico. A segunda vantagem é que toda a informação geográfica pode ser obtida e manipulada autonomamente, por qualquer outro cliente WMS/WFS, independentemente do *site* desenvolvido. Ou seja, se for necessário fazer algum tipo de análise, se for necessário sobrepor os recursos sobre outro tipo de informação geográfica, etc., tudo isto pode ser feito por uma ferramenta que seja conforme às especificações WMS/WFS.

Para todo o processo de manutenção/actualização do site decidiu-se usar um serviço WFS-T, dada a necessidade de alterar a componente geográfica (como já visto no capítulo anterior, na secção 3.2.1.1.).

Depois de algum tempo a investigar e testar os diferentes sistemas quer servidores quer clientes, decidiu-se optar pelo *GeoServer* como servidor Geo-WS e o *Mapbuilder*[39] como cliente. Ainda houve uma versão sobre o *iGeoPortal*[40], mas depois foi abandonada em detrimento do *Mapbuilder*. Foi também criado um cliente em PHP (apenas com algumas funcionalidades básicas como: zoom, pan e activar/desactivar camadas), bem como um cliente que usa a API do *Google Maps* para integração no site.

Toda a IG usada no site, inserida no *GeoServer*, foi carregada a partir de *shapefiles* devido ao facto de se ter feito um levantamento com as ferramentas da ESRI (com o *ArcPAD*[41] e *ArcMap*) e, principalmente, devido aos testes efectuados com o *GeoServer*, que se revelou sempre muito mais fiável e rápido a trabalhar com *shapefiles* do que com *MySQL* com extensões GIS.

## *4.2. A IG no GeoServer*

A interface gráfica de configuração do *GeoServer* via *Web* vem facilitar a questão da configuração dos serviços WFS e WMS. Na verdade, tudo que é feito graficamente traduz-se em alterações de ficheiros XML que contêm toda a configuração do *GeoServer*, portanto, pode-se alterar directamente esses ficheiros para produzir efeitos na configuração. Os ficheiros *service.xml* e *Catalog.xml* são os

[38] http://tikiwiki.org/
[39] http://mapbuilder.sourceforge.net/
[40] http://deegree.sourceforge.net/src/demos.html#client
[41] http://www.esri.com/software/arcgis/arcpad/index.html

dois ficheiros centrais da configuração (localizados na directoria ***WEB-INF/***), sendo que o primeiro contém as opções de configuração do servidor (como o número máximo de *features* a retornar, níveis de acesso, administrador do sistema e respectiva *password*, etc.), bem como a meta-informação para o WFS e o WMS. O segundo ficheiro guarda a configuração dos *DataStores*, *Namespaces* e *Styles*.


**Figura 49 – Iniciação do *GeoServer*.**

Partindo deste princípio, mostrar-se agora como configurar o *GeoServer*, que se encontra instalado e disponível em *http://esri.di.uminho.pt:8888/GeoServer*, para disponibilizar IG.

### 4.2.1. Configuração do *GeoServer*

Seguindo o *link* "*Config*" na ferramenta de administração, exitem quatro opções para a configuração do *GeoServer*: *Server*, WFS, WMS (informação relativa ao servidor e aos serviços de mapas e *features*, as alterações aqui feitas ficarão guardadas no ficheiro *service.xml*, previamente relatado) e *DATA* (configuração de todos os dados geográficos que serão disponibilizados e que ficarão guardados no ficheiro *Catalog.xml*, também já falado).

Partindo do princípio que a configuração do servidor e dos serviços WFS e WMS não oferece grande ciência, e que o *GeoServer* já vem configurado de raiz para satisfazer a maior parte dos utilizadores, mostra-se agora como configurar IG.

#### 4.2.1.1. *Namespace*

Pode-se aqui atribuir um identificador, prefixo, único para a IG. É através de um *namespace* que se vai distinguir os diferentes âmbitos da IG e aceder aos dados que se pretende realmente visualizar. Por exemplo, no caso prático do site *turismonoave.com*, existem vários concelhos e para cada concelho existem os mesmos requisitos geográficos a disponibilizar, portanto, para se distinguir as estradas da Trofa das estradas de Fafe cria-se um *namespace* para cada concelho e depois acede-se à informação do tipo *trofa:estradas* ou *fafe:estradas* conforme o pretendido.

Assim, e para o este exemplo, cria-se um novo *namespace* "ave" da seguinte forma:

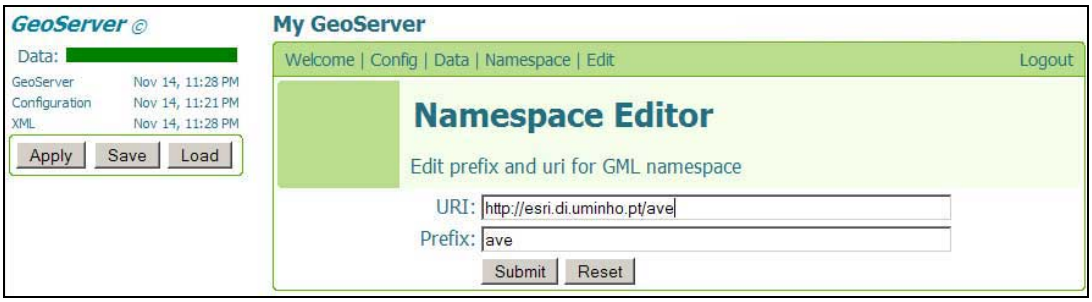



**Figura 50 – Criação de *namespaces* em *GeoServer*.**

**Nota:** A URI introduzida pode ou não ser válida, isto é, pode existir ou não informação relativa ao contexto da IG a apresentar. No entanto, neste campo, é obrigatório começar com a *string* "*http://*".

#### 4.2.1.2. *DataStore*

Nesta secção são indicadas as fontes dos dados que contêm a IG. Neste exemplo, a IG está guardada em *shapefiles* e é necessário definir dois *DataStores* (dado que existem duas *shapefiles* com a informação relativa aos concelhos do Ave e ao Rio propriamente dito).

Assim, para os concelhos cria-se um novo *DataStore* "concelhosAve" da seguinte forma:

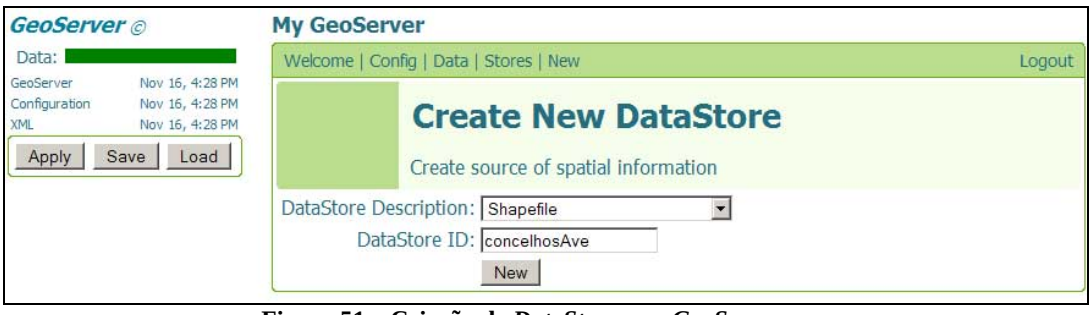



**Figura 51 – Criação de *DataStores* em *GeoServer*.**

Depois de escolher qual o *DataStore* e o seu respectivo nome, passa-se à fase de configuração desse *DataStore* (associando um *namespace*, uma descrição e uma localização) da seguinte forma:

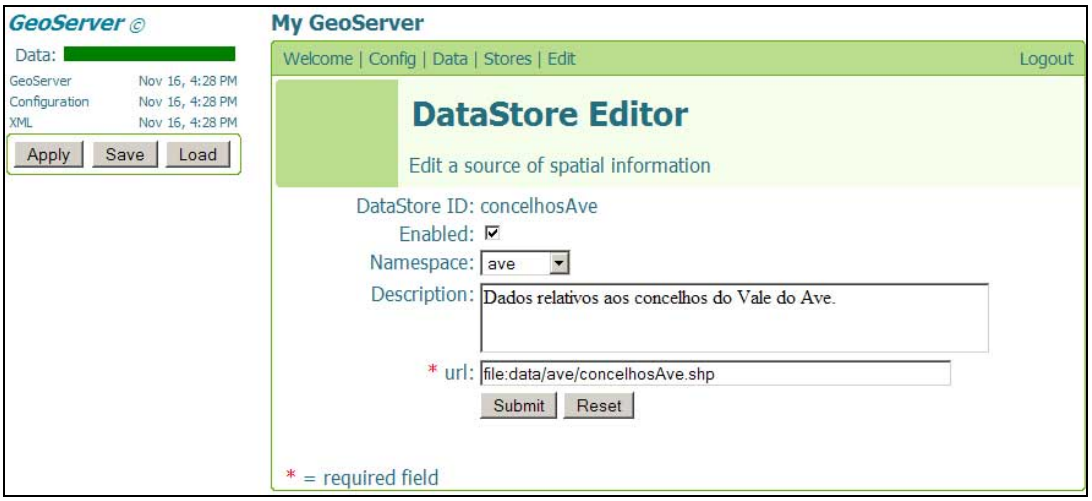



**Figura 52 – Configuração de *DataStores* em *GeoServer*.**

**Nota:** Na descrição da localização (campo URL) deve-se indicar uma localização a partir da directoria onde está o *GeoServer* instalado no servidor. Isto implica primeiro ter de fazer *upload* das *shapefiles* para o servidor. Não se conseguiu fazer funcionar o *GeoServer* em boas condições indicando um URL fora do servidor onde está instalado o *GeoServer*.

Da mesma forma como foi feito para os concelhos, configura-se um *DataStore* para o Rio Ave cuja informação está guardada em "*rio_ave.shp*" e fica-se assim com as duas fontes de dados geográficos para este exemplo.

### 4.2.1.3. *Styles*

No *GeoServer* tem-se a possibilidade de definir estilos, para uso futuro, para cada entidade na secção *Styles*. Para isso, basta dar um nome ao estilo que se vai usar e indicar qual o ficheiro SLD a carregar para o servidor. Existe também a opção, não obrigatória, para validar o ficheiro a carregar através do *Schema* SLD. Para este exemplo, criaram-se três ficheiros SLD: um responsável pelo desenho do rio, outro para o desenho de todos os concelhos e um outro que define como desenhar o nome de cada concelho.

**Nota:** Se se carregar um novo ficheiro SLD, este não poderá existir previamente no servidor.


**Figura 53 – Criação de estilos em *GeoServer*.**

### 4.2.1.4. *FeatureType*

Nesta secção são definidos os tipos de entidade a usar, ou seja, é nesta secção que se define toda a informação relativa a cada entidade. Nas secções anteriores define-se todo um conjunto de especificidades que vão ser usadas nesta secção para efectivamente caracterizar uma entidade.

Assim, começando por escolher um novo *FeatureType* tem-se de escolher primeiro qual o *DataStore* que este vai usar.

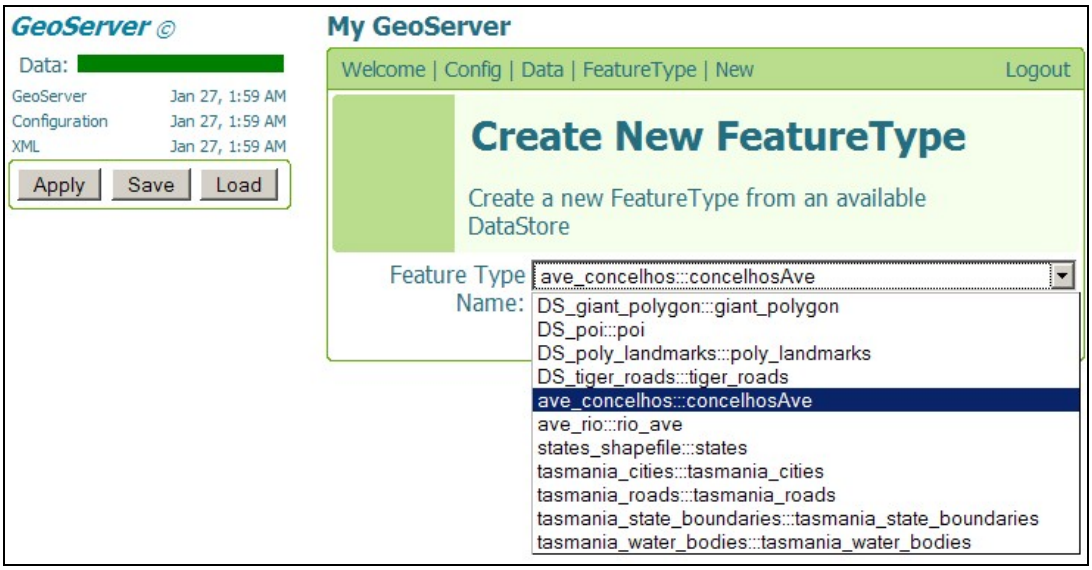



**Figura 54 – Criação de entidades em *GeoServer*.**

De seguida, aparece uma nova página de configuração em que são pedidas todas as características que faltam para que a entidade fique disponível, tais como: o tipo do estilo associado (criado na secção anterior), o sistema de coordenadas (para os exemplos aqui descritos usa-se o SRS 27492 que é o equivalente ao *Datum*73), as coordenadas referentes aos limites da entidade (aqui, se não se souber ao certo, pode-se clicar em "*Generate*". Desta forma, o *GeoServer* vai automaticamente consultar o *DataStore* e depois é só copiar os valores para os campos respectivos), informação abstracta e qual o esquema para a entidade.


**Figura 55 – Configuração de entidades em *GeoServer*.**

Da mesma forma que foi criado um *FeatureType* para os concelhos também se cria um para o rio.

Com o final desta fase, pode-se aplicar as alterações e guardar o estado do servidor (clicando em "*Apply*" e "*Save*" respectivamente e por esta ordem) e assim fica-se já com informação guardada e disponível para satisfazer pedidos de mapas.

Para o caso concreto do site *turismonoave.com*, para além desta informação, que se acaba de mostrar como se configura, tem-se de disponibilizar a seguinte informação para cada concelho:

- Mapa com a delimitação do concelho e das freguesias;
- Rios;
- Edifícios;
- Estradas;
- Ruas;
- Recursos Turísticos.

### 4.2.1.5. Teste da configuração do servidor

Depois de configurada e guardada a informação pretendida, pode-se fazer um pedido ao WMS (codificado com o método GET do protocolo HTTP) para visualizar essa mesma informação em qualquer *browser*, da seguinte forma:

```
http://localhost:8888/GeoServer/wms?SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap
&FORMAT=image/jpeg&BBOX=-
54838.8934320228,174384.07299711034,12627.089555286893,225712.48910300783&WIDTH=
500&HEIGHT=500&EXCEPTIONS=application/vnd.ogc.se_inimage&LAYERS=ave:concelhosA
ve.ave:concelhosAve.ave:rio  ave&SRS=EPSG:27492&STYLES=concelhos.nomeconc.rios
```

**Figura 56 – Pedido ao servidor WMS (configurado no *GeoServer*).**

A resposta do servidor será uma imagem no formato JPG da seguinte forma:

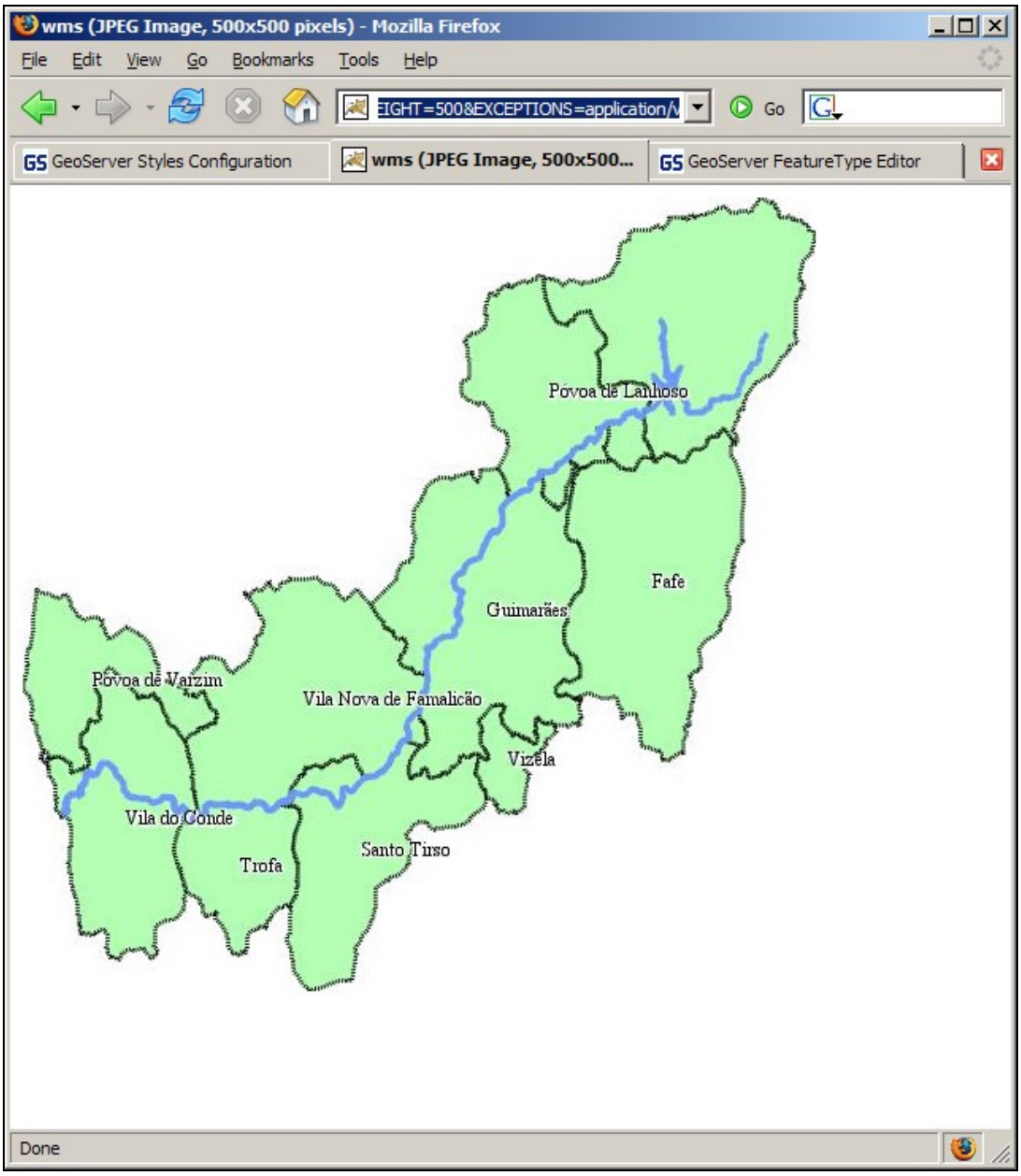



**Figura 57 – Resposta a um pedido ao servidor WMS (configurado no *GeoServer*).**

Um pedido ao WFS seria tipo:

```
http://localhost:8888/GeoServer/wfs?request=getfeature&service=wfs&version=1.1.1
&typename=ave:rio_ave
```

**Figura 58 – Pedido ao servidor WFS (configurado no *GeoServer*).**

E a respectiva resposta é um ficheiro GML tal como:

```
<wfs:FeatureCollection xsi:schemaLocation="http://www.opengis.net/wfs
http://localhost:8888/geoserver/schemas/wfs/1.0.0/WFS-basic.xsd http://localhost/ave
http://localhost:8888/geoserver/wfs/DescribeFeatureType?typeName=ave:rio_ave">
  <gml:boundedBy>
    <gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#27492">
      <gml:coordinates decimal="." cs="," ts=" ">
        -51545.77943534,185262.6761919 8492.71136685,217864.2767449
      </gml:coordinates>
    </gml:Box>
  </gml:boundedBy>
  <gml:featureMember>
    <ave:rio_ave fid="rio_ave.1">
      <ave:the_geom>
        <gml:MultiLineString srsName="http://www.opengis.net/gml/srs/epsg.xml#27492">
          <gml:lineStringMember>
            <gml:LineString>
              <gml:coordinates decimal="." cs="," ts=" ">
-23647.20976395,189148.34101408 -23655.72111246,189130.10853024 -23657.87060281,189127.13947684
-23694.24802625,189092.69547714 -23713.25443837,189068.92628612 -23726.0908446,189060.62634763
-23751.51220021,189048.96620064 -23774.72911749,189036.02362374 -23793.67778767,189017.13455814
-23811.87022468,188969.86253411 -23826.0663605,188946.28333271 -23828.11061151,188944.81743244
-23833.7208931,188943.42883187 -23845.11933871,188935.27231691 -23860.11175454,188930.56076071
-23890.05560806,188926.96027123 -23916.49676087,188916.98861038 -23944.71208142,188911.42023827
-23970.57860877,188904.08701164 -23995.41695704,188892.20800406 -24015.6815847,188876.79463084
-24017.98288048,188874.45794484
              </gml:coordinates>
            </gml:LineString>
          </gml:lineStringMember>
        </gml:MultiLineString>
      </ave:the_geom>
      <ave:OBJECTID>1</ave:OBJECTID>
      <ave:Entity>Polyline</ave:Entity>
      <ave:Handle>1C2</ave:Handle>
      <ave:Layer>51</ave:Layer>
      <ave:Color>5</ave:Color>
      <ave:Linetype>CONTINUOUS</ave:Linetype>
      <ave:Elevation>0.0</ave:Elevation>
      <ave:Thickness>0.0</ave:Thickness>
      <ave:Text_> </ave:Text_>
      <ave:Shape_Leng>487.445595197</ave:Shape_Leng>
    </ave:rio_ave>
  </gml:featureMember>
  ...
```

Text | Grid | Schema/WSDL | Authentic | Browser

**Figura 59 - Resposta a um pedido ao servidor WFS (configurado no *GeoServer*).**

## *4.3. Visualização de IG*

No contexto da visualização dos dados geográficos, existem já vários clientes que permitem uma visualização mais avançada que os simples pedidos ao *browser* (como já visto e que retorna apenas uma imagem estática da área geográfica pedida). Das aplicações testadas dá-se destaque a duas exaustivamente testadas: o *iGeoPortal*, da equipa do projecto *deegree*, e o *Mapbuilder*, que vem integrado com o *GeoServer*. Para além destas, foi criada de raiz uma *script* PHP para integração no site e tendo

como propósito mostrar a IG disponibilizada com algumas operações básicas (*pan*, *zoom* e sobreposição de camadas).

### 4.3.1. *Script* PHP para visualização básica

Devido a um requisito para o site, foi feita uma *script* PHP que lança pedidos automáticos ao servidor de mapas baseados na escolha do utilizador. Toda a interacção se baseia numa interface *Web* que, dado um concelho inicial, possibilita ao utilizador escolher as camadas (de IG) disponíveis para esse concelho, bem como fazer operações de visualização de imagens como zoom e *pan*.

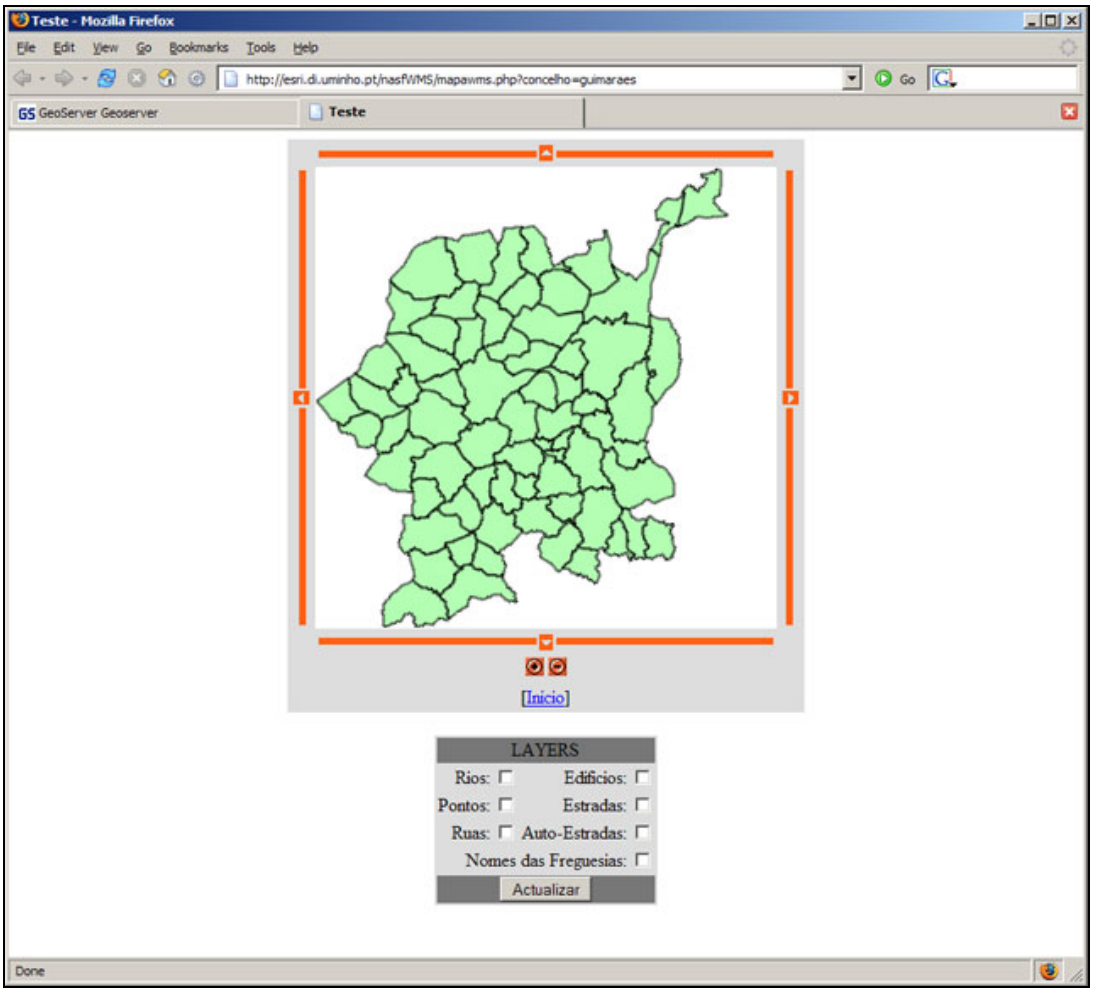



**Figura 60 – Script PHP para visualização de IG.**

A *script* funciona com base em instruções HTTP POST e GET e todos os valores para as operações são guardados em sessões. Desta forma, e dado um concelho (se não se incluir um concelho é mostrado um mapa do ave), irão ser guardadas as coordenadas dos cantos relativos à área desse mesmo concelho (coordenadas geográficas em Datum73 predefinidas para cada concelho).

```php
if (empty($_REQUEST['concelho'])) {
    $mun = 'ave';
    $min_x_ini = -55000;
    $min_y_ini = 174000;
    $max_x_ini = 13000;
    $max_y_ini = 242000;
} else {
    if ($_REQUEST['concelho']=='vminho'){
        $mun = 'vminho';
        $min_x_ini = -10550;
        $min_y_ini = 208500;
        $max_x_ini = 12750;
        $max_y_ini = 231800;
    }
    if ($_REQUEST['concelho']=='planhoso'){
        $mun = 'planhoso';
        $min_x_ini = -17610;
        $min_y_ini = 205475;
        $max_x_ini = -1145;
        $max_y_ini = 221940;
    }
```

**Figura 61 – Código PHP para inicialização da *script.***

Inicialmente também é guardado o nível de zoom (por defeito é atribuído o valor 1). A partir daí tem-se informação relevante para se poder fazer um pedido ao WMS que retorna uma imagem do concelho escolhido.

### 4.3.1.1. Operações

Para mostrar uma determinada área geográfica, já se sabe que se tem de fazer um pedido ao WMS com as coordenadas relativas aos cantos inferior esquerdo e superior direito dessa área. Transpondo isto para um referencial tem-se algo do tipo:

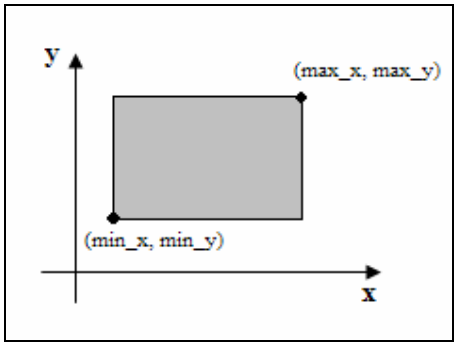



**Figura 62 – Representação, num referencial, de uma área geográfica.**

Portanto, todas as operações suportadas nesta *script* serão aplicadas aos pontos desses mesmos cantos assinalados.

#### *4.3.1.1.1. ZOOM*

O objectivo desta operação é aumentar ou diminuir (*ZoomIn* ou *ZoomOut*, respectivamente) o detalhe do mapa que está a ser disponibilizado. Isto quer dizer que, para uma operação de *Zoom*, tem de se fazer um novo pedido ao WMS, baseado no anterior, mas em que a área a ser mostrada será maior (menos detalhe, *ZoomOut*) ou menor (mais detalhe, *ZoomIn*). Para isso, tem de se encontrar as coordenadas do ponto central do mapa, calcular a distância do centro a cada um dos lados (pois a área a visualizar pode ser expressa num rectângulo), multiplicar essa distância pelo factor de Zoom (maior que 1 aumenta a área de visualização entre 0 e 1 diminui) e calcular os novos pontos baseados no resultado anterior.

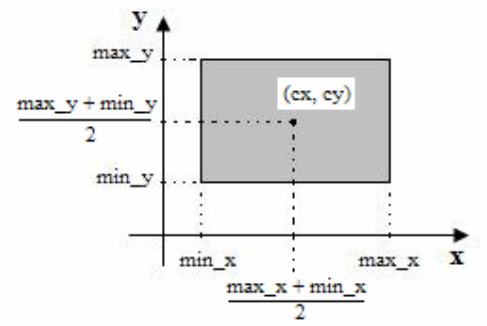
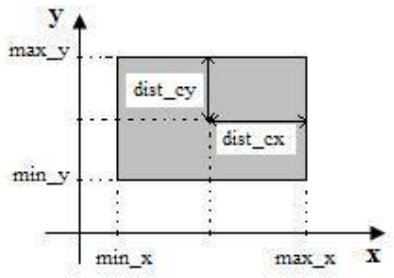
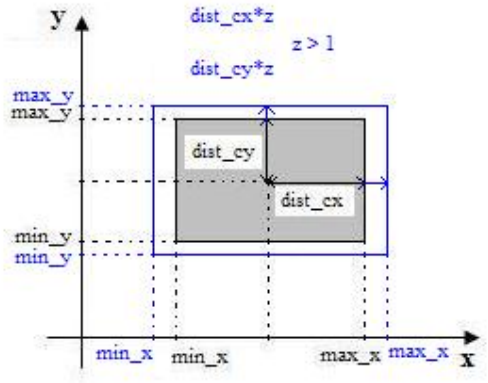
| Gráfico | Equação |
|---|---|
|  | $cx = \frac{\max\_x + \min\_x}{2}$<br>$cy = \frac{\max\_y + \min\_y}{2}$ |
|  | $dist\_cx = \max\_x - cx$<br>$dist\_cy = \max\_y - cy$ |
|  | $\min\_x = cx - (dist\_cx \times z)$<br>$\max\_x = cx + (dist\_cx \times z)$<br>$\min\_y = cy - (dist\_cy \times z)$<br>$\max\_y = cy + (dist\_cy \times z)$ |

**Figura 63 – Representação da operação de *Zoom*.**

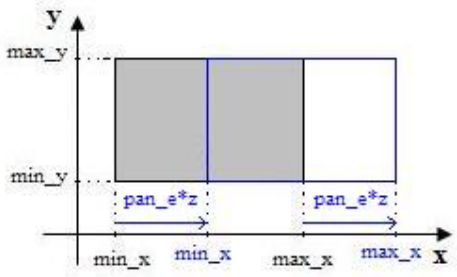
Então, após estes passos, tem-se os novos cantos referentes à área que se quer mostrar e serão guardados para posteriormente se proceder ao pedido ao WMS.

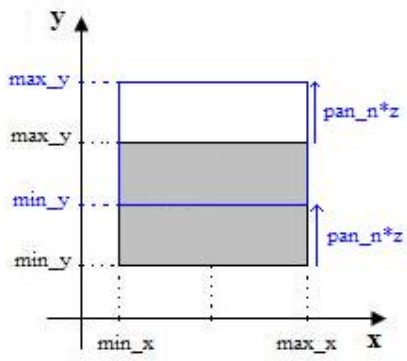
```
139     if (!empty($_REQUEST['zoom'])) {
140       $center_x = ($_SESSION['smax_x'] + $_SESSION['smin_x']) / 2;
141       $center_y = ($_SESSION['smax_y'] + $_SESSION['smin_y']) / 2;
142       $dist_cx = $_SESSION['smax_x'] - $center_x;
143       $dist_cy = $_SESSION['smax_y'] - $center_y;
144       $min_y = $center_y - ($dist_cy * $_REQUEST['zoom']);
145       $max_y = $center_y + ($dist_cy * $_REQUEST['zoom']);
146       $min_x = $center_x - ($dist_cx * $_REQUEST['zoom']);
147       $max_x = $center_x + ($dist_cx * $_REQUEST['zoom']);
148       $_SESSION['smin_x'] = $min_x;
149       $_SESSION['smax_x'] = $max_x;
150       $_SESSION['smin_y'] = $min_y;
151       $_SESSION['smax_y'] = $max_y;
152       $_SESSION['zoomfactor'] = $_SESSION['zoomfactor'] * $_REQUEST['zoom'];
153     }
```

**Figura 64 – Código PHP para operação de *Zoom*.**

### *4.3.1.1.2. PAN*

Neste tipo de operações pretende-se "deslocar" a área de visualização para uma nova orientação definida. Esta *script* permite deslocações nas direcções norte, sul, este e oeste, e tem em atenção o nível de *zoom* (quanto maior o nível de zoom menor a deslocação, pois faz sentido que, por exemplo, estando a visualizar-se um concelho a deslocação seja de 1000 em 1000 metros, enquanto que estando a visualizar-se um lugar ou urbanização a deslocação seja de 25 em 25 metros). Também como para a operação de *Zoom*, tem de se calcular novos pontos para a área a visualizar baseados num estado anterior. Para isso, tem de se saber que tipo de deslocação se trata (norte, sul, este ou oeste) e deslocar a área de visualização, no eixo respectivo, determinadas unidades (unidades que serão multiplicadas pelo nível de *zoom* actual).

| **Gráfico** | **Equação** |
|---|---|
| Deslocamento no eixo dos xx (este e oeste) | |
|  | $\min\_x = \min\_x + (pan\_e \times z)$<br>$\max\_x = \max\_x + (pan\_e \times z)$ |
| Deslocamento no eixo dos yy (norte e sul) | |




$$\min\_y = \min\_y + (pan\_n \times z)$$
$$\max\_y = \max\_y + (pan\_n \times z)$$

**Figura 65 - Representação da operação de *Pan*.**

Naturalmente que para os deslocamentos para Oeste e Sul a fórmula é a mesma exceptuando o sinal que passa a ser "-".

```
115     if (!empty($_REQUEST['pan_e'])) {
116       $min_x = $_SESSION['smin_x'] + ($_REQUEST['pan_e'] * $_SESSION['zoomfactor']);
117       $max_x = $_SESSION['smax_x'] + ($_REQUEST['pan_e'] * $_SESSION['zoomfactor']);
118       $min_y = $_SESSION['smin_y'];
119       $max_y = $_SESSION['smax_y'];
120       $_SESSION['smin_x'] = $min_x;
121       $_SESSION['smax_x'] = $max_x;
122     }
123     if (!empty($_REQUEST['pan_n'])) {
124       $min_y = $_SESSION['smin_y'] + ($_REQUEST['pan_n'] * $_SESSION['zoomfactor']);
125       $max_y = $_SESSION['smax_y'] + ($_REQUEST['pan_n'] * $_SESSION['zoomfactor']);
126       $min_x = $_SESSION['smin_x'];
127       $max_x = $_SESSION['smax_x'];
128       $_SESSION['smin_y'] = $min_y;
129       $_SESSION['smax_y'] = $max_y;
130     }
```

**Figura 66 - Código PHP para operação de *Pan*.**

Estas operações vão alterar apenas a área a visualizar. No entanto, a *script* permite também adicionar/remover camadas. Essas camadas a visualizar, são guardadas num *array* à medida que são seleccionadas. A par de cada camada é também construído um *array* com os estilos correspondentes.

### *4.3.1.1.2. CENTRAR NUM PONTO*

A *script* permite também centrar o mapa num dado ponto. Para isso, atribuiu-se uma área de visualização de 1000 metros à volta do ponto dado e calcula-se o nível de *zoom* desta nova área de visualização relativamente à área do concelho. O nível de *zoom* é então a razão entre a distância dos cantos (superior e inferior) da nova área de visualização e a distância dos cantos da área de visualização de todo o concelho respectivo.

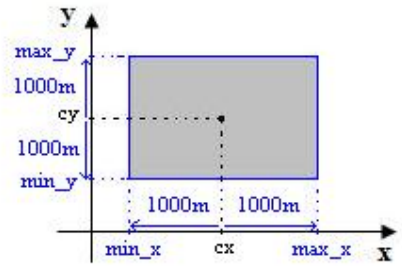
| Gráfico | Equação |
|---|---|
|  | $\min\_x = cx - 1000$<br>$\max\_x = cx + 1000$<br>$\min\_y = cy - 1000$<br>$\max\_y = cy + 1000$<br>$dist\_np = \sqrt{(\max x - \min x)^2 + (\max y - \min y)^2}$<br>$dist\_op = \sqrt{(\max x\_ini - \min x\_ini)^2 + (\max y\_ini - \min y\_ini)^2}$<br>$zoomfactor = \frac{dist\_np}{dist\_op}$ |



**Figura 67 - Representação da operação de centrar num ponto.**

```
154    if (!empty($_REQUEST['cx']) && !empty($_REQUEST['cy'])){
155      $min_x = $_REQUEST['cx'] - 1000;
156      $min_y = $_REQUEST['cy'] - 1000;
157      $max_x = $_REQUEST['cx'] + 1000;
158      $max_y = $_REQUEST['cy'] + 1000;
159      $dist_np = sqrt(pow(($max_x - $min_x),2) + pow(($max_y - $min_y),2));
160      $dist_op = sqrt(pow(($max_x_ini - $min_x_ini),2) + pow(($max_y_ini - $min_y_ini),2));
161      $_SESSION['zoomfactor'] = $dist_np / $dist_op;
162      $_SESSION['smin_x'] = $min_x;
163      $_SESSION['smax_x'] = $max_x;
164      $_SESSION['smin_y'] = $min_y;
165      $_SESSION['smax_y'] = $max_y;
166      $coordenadas = "&cx=".$_REQUEST['cx']."&cy=".$_REQUEST['cy'];
167    }
```

**Figura 68 - Código PHP para operação de centrar num ponto.**

Finalmente, depois de se ter as coordenadas da área a visualizar e as camadas pretendidas, é construído o pedido ao WMS baseado nos requisitos anteriores e em variáveis predefinidas (como o servidor, a versão do serviço,...).

```
173    $aLayers[0] = $mun.":freguesias";
174    $aStyles[0] = "freguesias";
175
176    if ($_REQUEST['edificios']){
177      $aLayers[1] = $mun.":edificios";
178      $aStyles[1] = "edificios";
179      $layers .= "&edificios=on";
180    }
181
182    if ($_REQUEST['ruas']){
183      $aLayers[2] = $mun.":ruas";
184      $aStyles[2] = "ruas";
185      $layers .= "&ruas=on";
186    }
```

**Figura 69 – Código PHP para construção das camadas a mostrar.**

```
$img_wms = "http://www.turismonoave.com:8888/geoserver/wms?service=WMS&VERSION=1.1.1&REQUEST=GetMap&";
$img_wms .= "SRS=EPSG:27492&BBOX=$min_x,$min_y,$max_x,$max_y&";
$img_wms .= "WIDTH=$width&HEIGHT=$height&FORMAT=image/png&BGCOLOR=0xfff8ff&TRANSPARENT=true&";
$img_wms .= "EXCEPTIONS=application/vnd.ogc.se_inimage";
$img_wms .= "&LAYERS=";

$strLayers = '';
for ($i=0; $i<7; $i++){
  if ($aLayers[$i])
    $strLayers .= $aLayers[$i].",";
}

$strlay = substr($strLayers,0,strlen($strLayers)-1);
$img_wms .= $strlay;

$strStyles = '';
for ($i=0; $i<7; $i++){
  if ($aStyles[$i])
    $strStyles .= $aStyles[$i].",";
}

$strsty = substr($strStyles,0,strlen($strStyles)-1);
$img_wms .= "&STYLES=".$strsty;
```

**Figura 70 – Código PHP para construção do pedido ao WMS.**

### 4.3.2. *iGeoPortal*

O *iGeoPortal* é o componente cliente/portal do projecto *deegree*. É um cliente *Web* (feito em JSP e *Javascript* e portanto preparado para correr em qualquer *browser*) e modular cuja configuração é baseada na especificação WMC. Diferentes módulos podem oferecer funcionalidade de cliente *Web Map*, assim como funções para clientes *gazetteer*, *catalog* ou WFS. Pode-se usar os módulos disponibilizados com o cliente mas também se podem criar novos módulos e integrá-los no cliente para satisfazer requisitos específicos. O *iGeoPortal* suporta também *layers* de um ou mais WMS e oferece a oportunidade de guardar o estado actual do cliente num documento XML em conformidade com *Web Map Context*. Para uma compreensão mais detalhada de todo o funcionamento e configuração deste portal, sugere-se uma leitura à documentação[42].

#### 4.3.2.1. Configuração

Quase toda a configuração do *iGeoPortal* é feita através de ficheiros XML e está muito bem documentada, portanto fica aqui apenas um exemplo de como foi configurado o cliente para mostrar informação relativa ao caso de estudo desta dissertação.

Os ficheiros de configuração estão todos alojados em "*$igeoportal_home$\WEB-INF*" e suas subdirectorias. Será necessário, eventualmente, alterar todos os ficheiros "*.xml*" e "*.xsl*" para ajustar correctamente o caminho destino da instalação do *iGeoPortal*.

O primeiro passo a dar é ir ao ficheiro "*$igeoportal_home$\WEB-INF\web.xml*" e definir qual o ficheiro "de arranque" do cliente (no valor do parâmetro *MapContext.configFile*). Ou seja, qual o ficheiro que contém a informação geográfica a ser mostrada a quando da primeira vez que entra no cliente. Esse

[42] http://deegree.sourceforge.net/src/igeoportal_docu.pdf

ficheiro, e assim como todos os outros que contêm dados geográficos, seguem a especificação WMC.

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
3     "http://java.sun.com/j2ee/dtds/web-app_2.2.dtd">
4   <web-app>
5     <servlet>
6       <servlet-name>RequestHandler</servlet-name>
7       <servlet-class>org.deegree_impl.clients.wmsclient2.control.MapRequestDispatcher</servlet-class>
8       <init-param>
9         <param-name>Handler.configFile</param-name>
10        <param-value>file:///E:/igeoportal/WEB-INF/xml/controller.xml</param-value>
11      </init-param>
12      <init-param>
13        <param-name>MapContext.configFile</param-name>
14        <param-value>E:/igeoportal/WEB-INF/xml/ave.xml</param-value>
15      </init-param>
16    </servlet>
17
18    <servlet-mapping>
19      <servlet-name>RequestHandler</servlet-name>
20      <url-pattern>/control</url-pattern>
21    </servlet-mapping>
22
23    <welcome-file-list>
24      <welcome-file>welcome.html</welcome-file>
25    </welcome-file-list>
26  </web-app>
```

**Figura 71 – Ficheiro para a configuração de arranque do *iGeoPortal*.**

Relativamente ao ficheiro "*ave.xml*", que é o que vai ser carregado logo de início, tem, como já se sabe, de seguir a especificação WMC. Aí tem de se descrever, obrigatoriamente, qual o tamanho da imagem, a área de visualização, alguma meta-informação sobre os dados e depois a lista das camadas a serem mostradas (com referência ao servidor WMS onde se encontra a IG guardada). Como por exemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<ViewContext version="1.0.0" id="ave" xmlns="http://www.opengis.net/context" xmlns:xlink="
http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
http://www.opengis.net/context http://schemas.opengis.net/context/1.0.0/context.xsd">
  <General>
    <Window width="500" height="500"/>
    <BoundingBox SRS="EPSG:27492" minx="-54838.8934320228" miny="174384.07299711034" maxx="
12627.089555286893" maxy="225712.48910300783"/>
    <Title>ave:rio_ave Map</Title>
    <KeywordList>
      <Keyword>ave:rio_ave</Keyword>
    </KeywordList>
    <Abstract></Abstract>
  </General>
  <LayerList>
    <Layer queryable="1" hidden="0">
      <Server service="OGC:WMS" version="1.1.1" title="ave:rio_ave Preview">
        <OnlineResource xlink:type="simple" xlink:href="http://localhost:8888/geoserver/wms"/>
      </Server>
      <Name>ave:rio_ave</Name>
      <Title>ave:rio_ave</Title>
      <SRS>EPSG:27492</SRS>
      <FormatList><Format current="1">image/png</Format></FormatList>
          <StyleList>
            <Style current="0">
              <Name>rios</Name>
              <Title>rios</Title>
            </Style>
          </StyleList>
    </Layer>
    <Layer queryable="1" hidden="0">
      <Server service="OGC:WMS" version="1.1.1" title="ave:concelhos Preview">
        <OnlineResource xlink:type="simple" xlink:href="http://localhost:8888/geoserver/wms"/>
      </Server>
      <Name>ave:concelhosAve</Name>
      <Title>ave:concelhosAve</Title>
      <SRS>EPSG:27492</SRS>
      <FormatList><Format current="1">image/png</Format></FormatList>
          <StyleList>
            <Style current="0">
              <Name>concelho</Name>
              <Title>concelho</Title>
            </Style>
          </StyleList>
    </Layer>
  </LayerList>
</ViewContext>
```

**Figura 72 – Ficheiro de contexto para um dado concelho.**

Assim, está pronta a configuração, note-se que é um ficheiro de configuração com o mínimo indispensável, e não fazendo uso da configuração dos módulos do *iGeoPortal* (para tal, sugere-se uma leitura atenta à documentação do cliente), para o *iGeoPortal* poder mostrar a IG que trouxemos para este caso de estudo. Então pode-se correr o cliente a partir de um qualquer *browser* (ex. *http://localhost:8888/igeoportal*) e tem-se a seguinte interface para a manipulação da IG pedida:

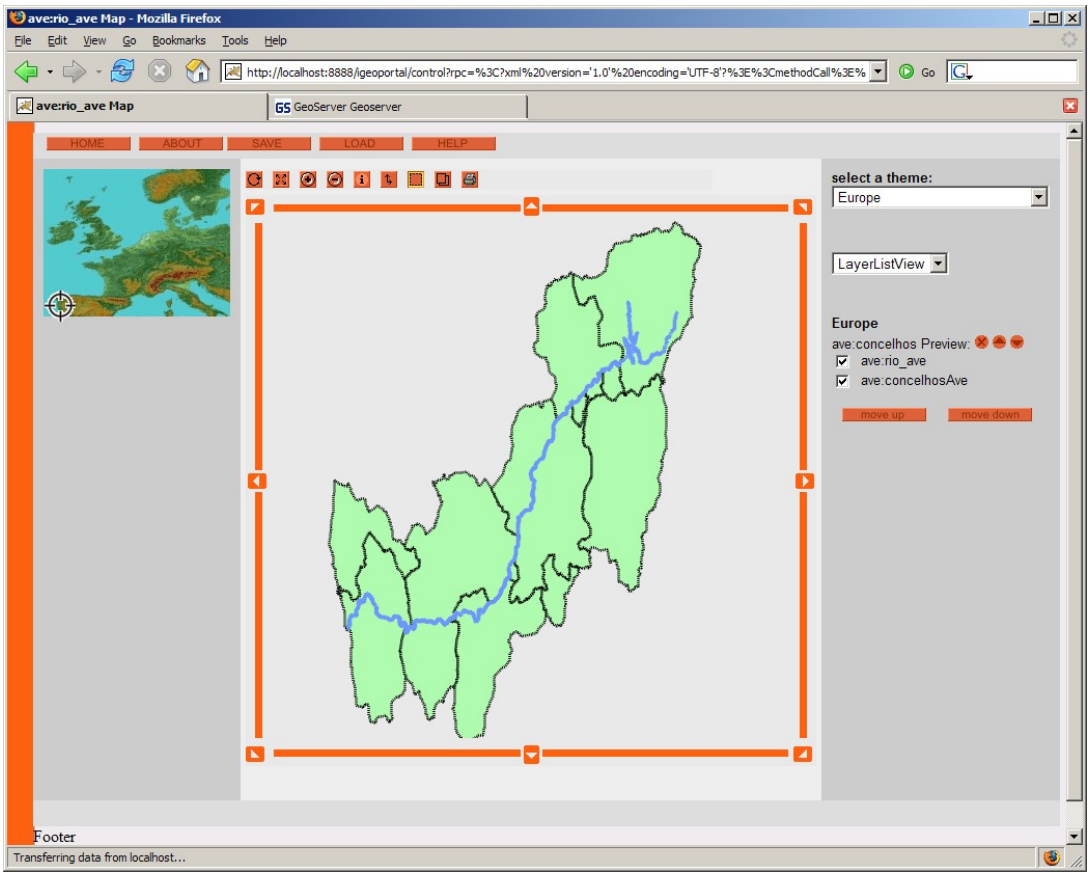



**Figura 73 – Visualização de IG no *iGeoPortal*.**

### 4.3.3. *MapBuilder*

O *MapBuilder* é um cliente *Web Mapping* que cumpre os standards OGC, permitindo o visionamento de mapas interactivos de várias fontes em páginas *Web* de serviços WMS ou WFS.

O *MapBuilder* usa tecnologia AJAX (basicamente é uma combinação das tecnologias XHTML, CSS, XML, XSLT, *XMLHttpRequest* e *Javascript*), corre em quase todos os *browsers* actuais fazendo uso de funções *Javascript* e também possibilita o uso/programação de novos módulos para estender as capacidades do cliente para usos específicos.

#### 4.3.3.1. Configuração

O *MapBuilder* corre como uma aplicação *Web*, portanto é necessário ter instalado um servidor *Web* (nestes exemplos, usamos o *Apache*). A partir daí, e como no *iGeoPortal*, o *Mapuilder* é configurável maioritariamente a partir de ficheiros XML. Os ficheiros que guardam o contexto da IG seguem a especificação WMC.

Para uma compreensão mais detalhada de todo o funcionamento e configuração do *MapBuilder*, sugere-se uma leitura à documentação[43]. Aqui apenas se descreve como incluir o *MapBuilder* numa página *Web* com as funções básicas de navegação em mapas.

Assim, tome-se como exemplo uma página para mostrar o caso do mapa do Ave com o rio. Então cria-se uma subdirectoria "ave" na directoria do *MapBuilder*. Aí criaram-se três ficheiros: um para a página *Web* ("*index.html*"), um contendo a configuração do *layout* do *MapBuilder* (*confAve.xml*) e um outro contendo a IG conforme WMC ("*Ave.xml*").

O ficheiro "*Ave.xml*" já foi visto na secção anterior como é descrito, portanto pode-se usar toda a informação lá disponível.

O ficheiro "*confAve.xml*" descreve a configuração para todas as funcionalidades (módulos) que se quer usar na apresentação da IG. Para este caso simplista configura-se esse ficheiro para disponibilizar zoom, *pan* e voltar ao estado inicial.

```
1   <?xml version="1.0" encoding="utf-8" standalone="no"?>
2   <MapbuilderConfig version="0.2.1" id="simpleTemplate" xmlns="
    http://mapbuilder.sourceforge.net/mapbuilder" xmlns:xsi="
    http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
    http://mapbuilder.sourceforge.net/mapbuilder ../../lib/schemas/config.xsd">
3     <models>
4       <Context id="mainMap">
5         <defaultModelUrl>Ave.xml</defaultModelUrl>
6         <widgets>
7           <MapPane id="mainMapWidget">
8             <htmlTagId>mainMapPane</htmlTagId>
9             <mapContainerId>mainMapContainer</mapContainerId>
10          </MapPane>
11          <Loading id="loading"/>
12          <AoiBoxDHTML id="aoiBox2">
13            <htmlTagId>mainMapPane</htmlTagId>
14            <stylesheet>../lib/widget/Null.xsl</stylesheet>
15            <mapContainerId>mainMapContainer</mapContainerId>
16            <lineColor>#FF0000</lineColor>
17            <lineWidth>1</lineWidth>
18            <crossSize>15</crossSize>
19          </AoiBoxDHTML>
20        </widgets>
21        <tools>
22          <AoiMouseHandler id="mainAoi"/>
23          <DragPanHandler id="mainDragPan">
24            <enabled>false</enabled>
25          </DragPanHandler>
26        </tools>
27      </Context>
28    </models>
```

**Figura 74 – Ficheiro de configuração inicial para o *MapBuilder* (1º parte).**

Assim, nesta primeira parte define-se qual o ficheiro WMC a usar (linha 5), a classe responsável por desenhar a IG (linhas 7 a 10), uma função de espera para o carregamento da página (linha 11), uma moldura para o mapa (IG, linhas 12 a 19) e os eventos disponibilizados (linhas 21 a 26).

[43] http://mapbuilder.codehaus.org

```
29  <widgets>
30    <ZoomIn id="zoomIn">
31      <buttonBar>mainButtonBar</buttonBar>
32      <targetModel>mainMap</targetModel>
33      <mouseHandler>mainAoi</mouseHandler>
34      <class>RadioButton</class>
35      <selected>true</selected>
36      <enabledSrc>/images/ZoomInEnable.png</enabledSrc>
37      <disabledSrc>/images/ZoomInDisable.png</disabledSrc>
38    </ZoomIn>
39    <ZoomOut id="zoomOut">
40      <buttonBar>mainButtonBar</buttonBar>
41      <targetModel>mainMap</targetModel>
42      <mouseHandler>mainAoi</mouseHandler>
43      <class>RadioButton</class>
44      <enabledSrc>/images/ZoomOutEnable.png</enabledSrc>
45      <disabledSrc>/images/ZoomOutDisable.png</disabledSrc>
46    </ZoomOut>
47    <DragPan id="dragPan">
48      <buttonBar>mainButtonBar</buttonBar>
49      <targetModel>mainMap</targetModel>
50      <mouseHandler>mainDragPan</mouseHandler>
51      <class>RadioButton</class>
52      <enabledSrc>/images/PanEnable.png</enabledSrc>
53      <disabledSrc>/images/PanDisable.png</disabledSrc>
54    </DragPan>
55    <Reset id="reset">
56      <buttonBar>mainButtonBar</buttonBar>
57      <targetModel>mainMap</targetModel>
58      <class>Button</class>
59      <disabledSrc>/images/ResetExtentDisable.png</disabledSrc>
60    </Reset>
61  </widgets>
62  <skinDir>../lib/skin/default</skinDir>
63  <widgetTextUrl>widgetText.xml</widgetTextUrl>
64  <!-- relative to the skin dir -->
65 </MapbuilderConfig>
```

**Figura 75 - Ficheiro de configuração inicial para o *MapBuilder* (2º parte).**

Nesta segunda parte, define-se quais os botões que se quer usar e acções/eventos respectivos (linhas 29 a 61), a pasta que contém os ficheiros CSS para o *layout* (linha 62) e ficheiro com os textos associados aos botões (linha 63).

Finalmente, no ficheiro "*index.html*" define-se o *layout* da apresentação da IG.

```
1   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
2   <html>
3    <head>
4     <title>Mapbuilder Simple Demo</title>
5     <link rel='StyleSheet' type='text/css' href='../lib/skin/default/docsStyle.css'>
6     <link rel='StyleSheet' type='text/css' href='../lib/skin/default/mapStyle.css'>
7     <script>
8      // URL of Mapbuilder configuration file.
9      var mbConfigUrl='confAve.xml';
10
11    </script>
12    <script type="text/javascript" src="../lib/Mapbuilder.js"></script>
13   </head>
14   <body onload="mbDoLoad()">
15    <h1><a href="http://mapbuilder.sourceforge.net">MapBuilder</a> Simple Demo</h1>
16    <div id="mainMapPane">
17     <noscript>
18      this page requires Javascript to be enabled
19     </noscript>
20    </div>
21    <div id="mainButtonBar">
22    </div>
23    <div id="loading">
24     <p>Loading Program<br/>
25     <img src="../lib/skin/default/images/Loading.gif"/>
26     </p>
27    </div>
28   </body>
29  </html>
```

**Figura 76 – Ficheiro de apresentação do *MapBuilder*.**

Então, tem de se definir quais os ficheiros CSS (linhas 5 e 6), o ficheiro de configuração do *MapBuilder* a ser guardado numa variável *Javascript* (linhas 7 a 11), a classe principal com as funções *Javascript* do *MapBuilder* (linha 12) e depois os módulos que se querer usar: painel para o desenho do mapa (linhas 16 a 20), uma barra de botões (linhas 21 a 22) e um modulo de espera pelo carregamento da página (linhas 23 a 26).

A função "*mbDoLoad()*" na *tag body* é obrigatória, pois inicia todos os processos do *MapBuilder*.

Depois de criados estes ficheiros, pode-se ir a uma janela do *browser* e arrancar o "*index.html*".

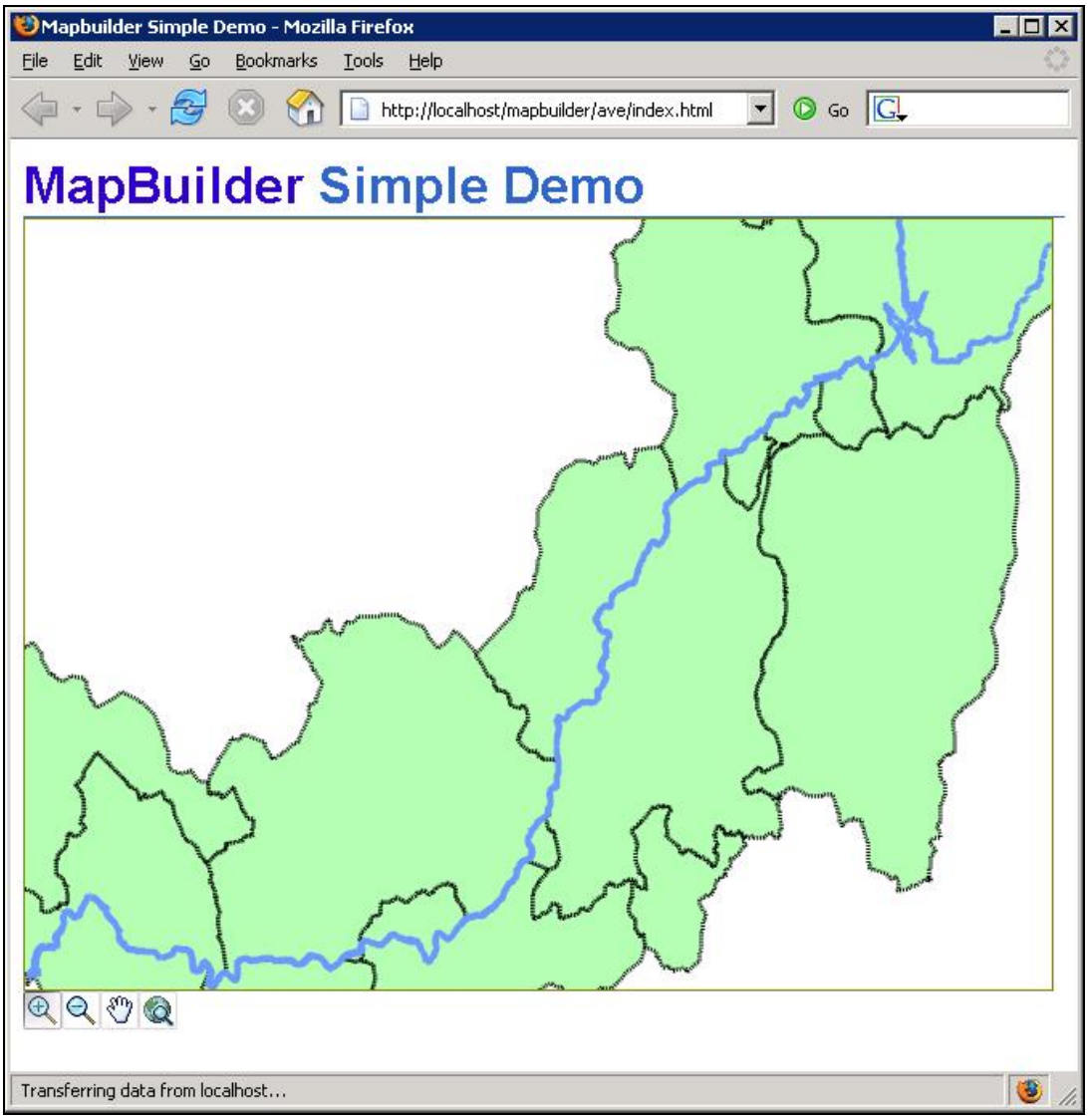



**Figura 77 - Visualização de IG no *MapBuilder*.**

Para o site *turismonoave.com*, decidiu-se optar, como cliente para visualizar a IG, pelo *MapBuilder* dado que este se revelou sempre muito mais eficiente em comparação com o *iGeoPortal*. Dado usar tecnologia AJAX, quase todas as acções do *MapBuilder* são feitas no cliente dependendo apenas do servidor para processar e disponibilizar o pedido ao WMS. Já o *iGeoPortal* usa muitas bibliotecas feitas em Java (*deegree*, *xalan*, *xerces*, *batik*, etc.), o que sobrecarrega o servidor ao processar essas bibliotecas. Numa aplicação SIG, integrada num site que será usado por muitos utilizadores em simultâneo, é natural que quanto menos ocupado o servidor estiver mais rápidos os pedidos serão disponibilizados.

### 4.3.4. Usando a API do *Google Maps*

No site *turismonoave.com* usa-se a API do *Google Maps* para mostrar os mapas de satélite dos recursos disponíveis. Assim, disponibiliza-se, através de uma *script* PHP, para um dado recurso turístico, o mapa de satélite da zona envolvente

com uma legenda (com o nome do recurso), o respectivo ícone e as operações de zoom e *pan*.

Como seria de esperar, o sistema de coordenadas usado no *Google Maps* (Coordenadas Geográficas, WGS84) é diferente do usado nas *shapefiles* que guardam a IG (Datum73 de Lisboa) no caso de estudo apresentado. Por isso, antes de serem passadas as coordenadas à *script* que usa a API tem de se convertê-las de Datum73 para WGS84 e para isso é usada a aplicação PROJ 4.4.6 [44].

A *script* PHP que usa a API é a seguinte:

```
1  <?
2  echo "<html xmlns=\"http://www.w3.org/1999/xhtml\">";
3  echo "<head>";
4  echo "<script
.  src=\"http://maps.google.com/maps?file=api&v=1&key=ABQIAAAA28EUEh1FcHjo5L81Fq3OQxT3TO3GWqye7gjPM64M5
.  HNrtXl3oxR2tkJLYUZqwwkp8Z5Tru8yqIPWEw\" type=\"text/javascript\"></script>";
5  echo "</head>";
6  echo "<body>";
7  echo "<table width=\"100%\" align=\"center\"><tr><td><div id=\"map\" style=\"width: 500px; height:
.  400px\"></div></td></tr></table>";
8
9  echo "<script type=\"text/javascript\">";
10 echo "var icon = new GIcon();";
11 echo "icon.image = \"http://esri.di.uminho.pt/tiki-1.8/simbolos/png20/".$_REQUEST['icon'].".png\";";
12
13 echo "icon.iconSize = new GSize(20, 20);";
14 echo "icon.iconAnchor = new GPoint(12, 12);";
15 // Creates one of our tiny markers at the given point
16 echo "function createMarker(point) {";
17 echo "var marker = new GMarker(point, icon);";
18 echo "map.addOverlay(marker);}";
19
20 echo "var map = new GMap(document.getElementById(\"map\"),[G_SATELLITE_TYPE]);";
21 echo "map.addControl(new GSmallMapControl());";
22     //map.disableDragging();
23 echo "map.centerAndZoom(new GPoint(".$_REQUEST['lon'].", ".$_REQUEST['lat']."), 3);";
24 echo "createMarker(new GPoint(".$_REQUEST['lon'].", ".$_REQUEST['lat']."));";
25 echo "map.openInfoWindow(new GPoint(".$_REQUEST['lon'].", ".$_REQUEST['lat']."),
.  document.createTextNode(\"".$_REQUEST['titulo']."\"));";
26 echo "</script>";
27 echo "</body>";
28 echo "</html>";
29 ?>
```

**Figura 78 – *Script* para geração de uma página HTML contendo a API do *Google Maps*.**

O objectivo é gerar uma página HTML daí o comando PHP "*echo*" em cada linha. A definição da *script Javascript* a ser usada vem na linha 4 (aquando do pedido da chave é recebido um pedaço de código *Javascript* para ser incluído no *site*). Na linha 7, define-se uma área de visualização para o mapa recebido (a etiqueta "*<div id="map"…>*" é obrigatória para ser possível ver o mapa). Da linha 10 à 14, define-se o ícone a usar. Da linha 16 à 18, uma função para gerar um marcador sobre o mapa num ponto dado (esse marcador é o ícone). Na linha 20, define-se o mapa e o seu tipo (mapa de satélite). Na linha 21, adiciona-se um tipo de controlo ao mapa (para zoom e *pan*). Na linha 23, define-se o centro do mapa (coordenadas já convertidas do recurso). Na linha 25, é criada a legenda com o nome do recurso.

[44] http://www.remotesensing.org/proj

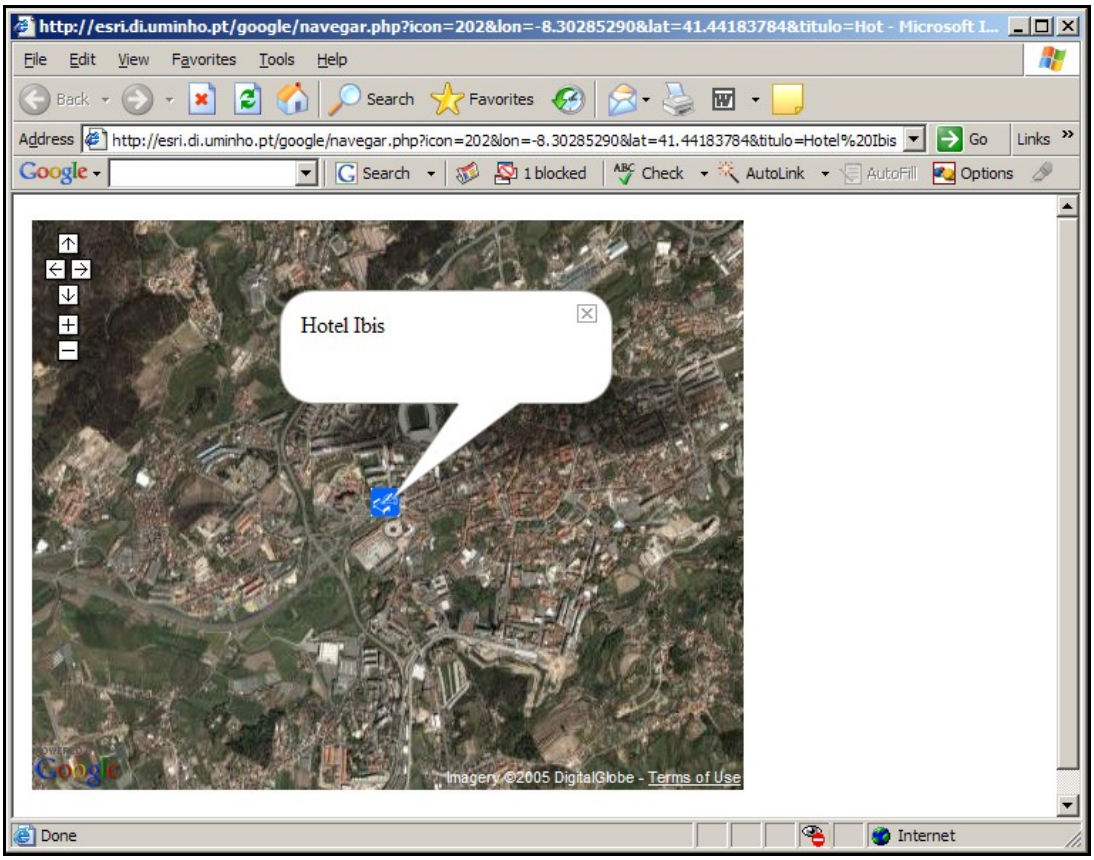



**Figura 79 – A API do *Google Maps* usada no site *turismonoave.com*.**

# 5. Actualização de IG na *Web*

No âmbito desta dissertação, procurou-se criar diferentes exemplos (provas de conceito) que fossem um bom ponto de partida para a concretização de todas estas ideias relacionadas com a actualização de IG via *Web*. Os três casos que neste capítulo serão apresentados derivam de diferentes abordagens e técnicas usadas para esse efeito.

Os dois primeiros casos estão directamente relacionados com o caso de estudo apresentado (site *turismonoave.com*), pois referem-se a soluções preconizadas para satisfazer requisitos pedidos para esse projecto. No primeiro caso, tem-se um exemplo de como mostrar IG (mapas) através de SVG e de como se pode manipular e actualizar coordenadas geográficas. Essas coordenadas são guardadas numa BD em campos para números reais e são alteradas através de *script* PHP. O modo como se chega a essas coordenadas é através de *Javascript* que é embutido no código SVG do mapa respectivo. No segundo caso, tem-se um caso típico de actualização de coordenadas geográficas através de pedidos a um WFS-T. A IG neste caso está disponível em *shapefiles* e a alteração é feita através de formulários.

O terceiro caso deriva de um interesse académico, inspirado pelo projecto Wikipédia, na tentativa de provar todas as teorias aqui apresentadas. É um caso típico de actualização de polígonos em BDs, que implementam extensões GIS, através de pedidos a um WFS-T. Faz-se uso do projecto *Mapbuilder* para recolher as coordenadas dos polígonos a alterar.

## *5.1. Localização de Recursos Turísticos*

No site *turismonoave.com*, aquando da edição de recursos turísticos, é dada a opção ao utilizador para localizar no mapa do respectivo concelho, o recurso que está a introduzir.


**Figura 80 – Atalho, no sito *turismonoave.com*, para o Mapa de auxílio.**

Desta forma, o utilizador poderá navegar no mapa do concelho à procura da localização exacta do recurso e, achando-a, basta clicar nesse local para automaticamente receber no formulário de edição as coordenadas geográficas do recurso.

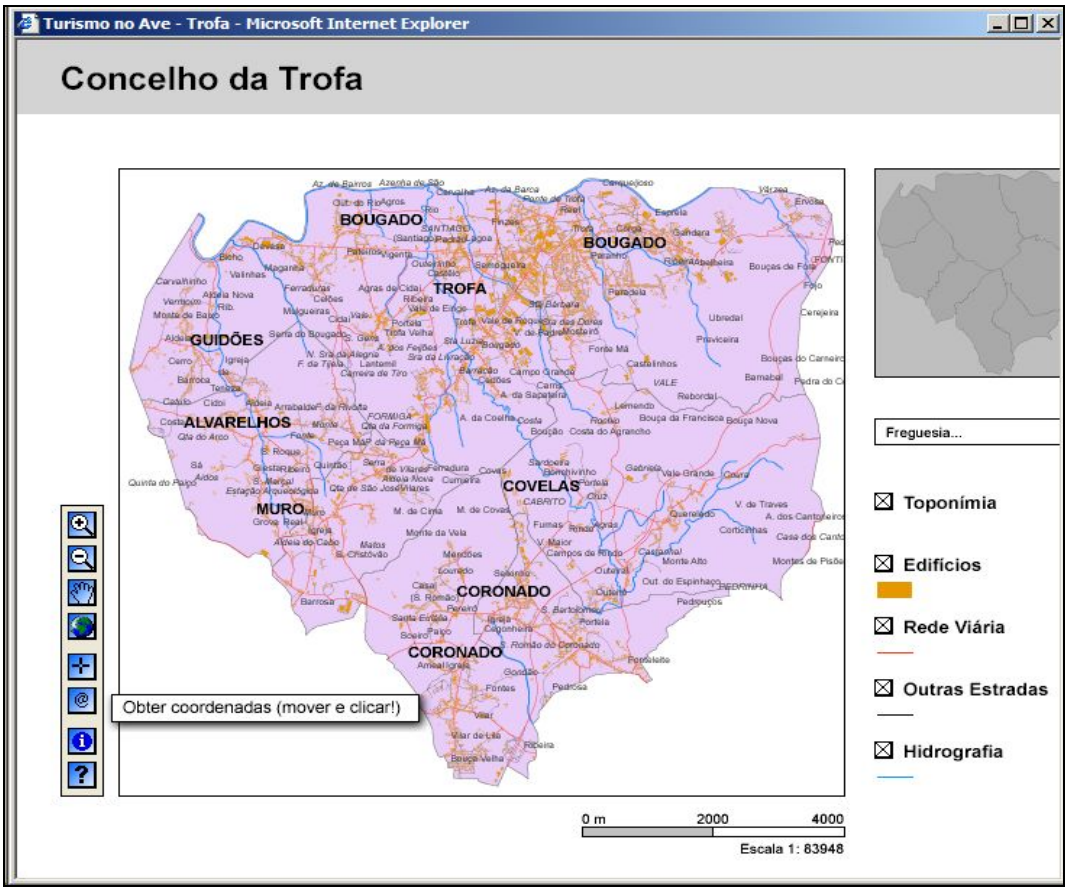



**Figura 81 – Mapa de auxílio para marcação de coordenadas.**

Estas coordenadas vão ser depois gravadas para a BD *MySQL* de suporte ao site, pois pretende-se ficar com o registo das coordenadas que já foram marcadas no mapa.

### 5.1.1. Geração de mapas em SVG

Os mapas dos concelhos que são disponibilizados para este efeito estão em SVG e foram gerados convertendo cada uma das *shapefiles* existentes para SVG. Além do SVG relativo à geometria das entidades geográficas, a interface recorre a um conjunto significativo de funções escritas em *Javascript*. Toda a interacção, ampliar/reduzir, mover, ajustar ao ecrã, marcar pontos, escolher freguesia, activar/desactivar tema, ajustar a escala e sincronização entre o mapa e o pequeno mapa de contexto é feita em *Javascript*. A sofisticação desta interface atesta, de alguma forma, o que já foi referido em relação à utilização do SVG, na secção 3.3.4.

### 5.1.2. Actualização de coordenadas

A integração destes mapas no site é feita através de *Javascript*. Assim, quando um utilizador pretende usar o mapa do concelho para localizar o local do recurso e receber as coordenadas dessa localização, poderá clicar em "Mapa de Auxílio" e, consequentemente, aparece uma janela nova com o mapa.

O código para tal execução é:

```
<input type=button value="{tr}Suport map{/tr}"
        onclick="javascript:saca_coordenadas();">
```

**Figura 82 – Código para abrir Mapa de Auxílio.**

A função *saca_coordenadas* vai detectar se o utilizador já tem um concelho escolhido para o recurso e, se afirmativo, vai abrir uma janela com o mapa do concelho escolhido.

```
function saca_coordenadas() {
var combo = document.getElementById('codconcelho');

if (combo[combo.selectedIndex].value == '6') {
        small_window('mapas/trofa/index.svgz');
} else {
        if (combo[combo.selectedIndex].value == '5') {
                small_window('mapas/stotirso/index.svgz');
        }
        ...
        else {
                alert('Tem que escolher um concelho');
        }
}
}

function small_window(myurl) {
var newWindow;
var props =
        'scrollBars=yes,resizable=yes,toolbar=no,menubar=no,location=n
        o,directories=no,width=720,height=620';
newWindow = window.open(myurl, "Georeferenciacao", props);
}
```

**Figura 83 – Código para função *saca_coordenadas*.**

Na nova janela aparece, então, um mapa em SVG (extensão .svgz[45]) com a informação geográfica de contexto existente para cada concelho. Esta inclui, além dos limites administrativos dos concelhos e freguesias, toda a rede viária, incluindo caminhos, a rede hidrográfica e os edifícios. Esta informação parece que é suficiente para identificar a localização dos recursos turísticos, sendo assumido que pode introduzir erros na casa das dezenas de metros.

[45] A extensão SVGZ significa que o ficheiro é em SVG, mas foi gravado com compactação em gzip. Os ficheiros SVG comprimidos são normalmente 50% a 80% mais pequenos do que os originais e são automaticamente descomprimidos pelos visualizadores, que o suportem, sem perdas significativas de performance.

Sobre estes mapas gerados, acrescenta-se um par de funcionalidades para permitir obter as coordenadas de um ponto marcado por um utilizador no mapa e também para mostrar no mapa todos os pontos já incluídos na base de dados. Para isso, tem-se na barra de ferramentas dois botões, para os quais são necessárias duas imagens (quatro na realidade, duas para cada acção para distinguir opção activa de não activa) e das respectivas acções em *Javascript*.

### 5.1.2.1. Marcar pontos

Para marcar um ponto no mapa e obter as suas coordenadas, criam-se então:

**1)** Imagens com os eventos associados para essa acção.

```
<image mvns:id="butOn" id="butkoordon"
        xlink:href="pictures/koordon.png"
        x="5" y="110"  width="20" height="20" visibility="hidden"
        onmouseover="toolTipOver(evt,'Obter coordenadas (mover e
                clicar!)');"
        onclick="clearEvents();" />

<image mvns:id="butOff" id="butkoordoff"
        xlink:href="pictures/koordoff.png"
         x="5" y="110" width="20" height="20" visibility="visible"
         onmouseover="toolTipOver(evt,'Obter coordenadas (mover e
                clicar!)');"
        onclick="arrangeEvents('koord');" />
```

**Figura 84 – Código para obter coordenadas de um ponto.**

**2)** Para os eventos associados a essa opção, tem de se activar a opção para mostrar as coordenadas à medida que o rato se desloca no mapa, criar um novo elemento SVG que vai agrupar todos os pontos que vão ser marcados, criar um elemento SVG (de acordo com a API do SVG, que é em todo idêntica à API do DOM para a manipulação de documentos XML) para cada ponto marcado (naturalmente sub elemento do grupo 'turismo' previamente criado, se ainda não existir) e passar as coordenadas dos pontos marcados para o formulário de edição.

```
function doShowCoordinates(evt) {
if (koordManagement) {
        var XWert = (theXOrigin + getEvtX(evt.clientX)).toFixed(thePrecision);
        var YWert = (theYOrigin - getEvtY(evt.clientY)).toFixed(thePrecision);
        var theText = XWert + ':' +  YWert;
// para retornar as coordenadas ao form da janela que pediu o mapa...
        browserEval('window.opener.addLongitude("' + theText + '")');
        SVGRoot.getElementById('measureText').firstChild.data = theText;

        if (!SVGmap.getElementById('turismo')) {
                var symbol = document.createElement('g');
                symbol.setAttribute("id",'turismo');
                SVGmap.appendChild(symbol);
        }


        var symbol = document.createElement('rect');
        symbol.setAttribute("x",getEvtX(evt.clientX)-40);
        symbol.setAttribute("y",getEvtY(evt.clientY)-40);
        symbol.setAttribute("width",80);
        symbol.setAttribute("height",80);
        symbol.setAttribute("rx",20);
        symbol.setAttribute("ry",20);
        symbol.setAttribute("style",'fill-opacity:0.5;fill:blue;pointer-
events:none;stroke:white;stroke-width:2;');
        symbol.setAttribute('visibility','visible');

        var x = SVGmap.getElementById('turismo');
        x.appendChild(symbol);
}
}
```

**Figura 85 – Código para a função *doShowCoordinates*.**

Quando o utilizador receber as coordenadas do recurso que está a editar pode gravar essa informação para a base de dados dos recursos.

### 5.1.2.2. Mostrar pontos

O objectivo desta funcionalidade é mostrar ao utilizador quais os recursos marcados já existentes na base de dados. Para isso, começa-se por criar uma imagem com eventos para essa operação:

```
<image mvns:id="butOff" id="butmeasureoff"
        xlink:href="pictures/getdata.png" x="5" y="135" width="20"
        height="20" visibility="visible"
        onmouseover="toolTipOver(evt,'Receber pontos já marcados no
servidor');"
        onclick="actualizaPontos();" />
```

**Figura 86 – Código para mostrar pontos guardados.**

A função *actualizaPontos* começa por limpar todos os pontos existentes no mapa, vai buscar os pontos existentes na BD (através da execução de uma *script* PHP) e adiciona esses pontos recebidos ao documento SVG (ao mapa).

```
function actualizaPontos() {
        var pedido = '';
        limpaPontos();
        pedido = 'http://localhost/turismonuave/turismo_svg.php?';
        pedido = pedido + 'codconcelho='+theConcelho+'&longitude='
                +theXOrigin+'&latitude='+theYOrigin;
        // pedido = 'trofa.svg'; // para testar localmente
        getURL(pedido, fileLoaded);
        function fileLoaded(data) {
                var string = ''; var ok=0;
                if(data.success) {
                        string = data.content;
                        ok=1;
                } else {
                        alert('A operação falhou.');
                        return;
                }
                var node = parseXML(string, document);
                if (!SVGmap.getElementById('turismo')) {
                        var symbol = document.createElement('g');
                        symbol.setAttribute("id",'turismo');
                        SVGmap.appendChild(symbol);
                }
                var x = SVGmap.getElementById('turismo');
                x.appendChild(node);
                var contador = 0;
                if (SVGmap.getElementById('turismo')) {
                        contador =
SVGmap.getElementById('turismo').getElementsByTagName('rect').length;
                }
                if (ok) {
                        alert('Informação sobre ' + contador + ' ponto(s) recebida
com sucesso');
                }
        }
}
```

**Figura 87 – Código para a função *actualizaPontos*.**

```
function limpaPontos() {
        if (SVGmap.getElementById('turismo')) {
                SVGmap.removeChild(SVGmap.getElementById('turismo'));
                var symbol = document.createElement('g');
                symbol.setAttribute("id",'turismo');
                SVGmap.appendChild(symbol);
        }
}
```

**Figura 88 – Código para a função *limpaPontos*.**

A *script* PHP, que vai buscar os pontos à BD, vai gerar um grupo de elementos SVG. Esse grupo é constituído por elementos "*rect*" SVG, que correspondem a cada ponto guardado na BD.

```
...
$data = $turismolib->get_svg($_REQUEST["codconcelho"]);
$type = 'image/svg+xml';
$pontos = ''; $etiquetas = '';
for ($i = 0; $i < count($data); $i++) {
        $res  = "<rect x=\"";
        $res .= $data[$i]["longitude"]-$longitude-40;
        $res .= "\" y=\"";
        $res .= $latitude - $data[$i]["latitude"]-40;
        $res .= "\" width=\"80";
        $res .= "\" height=\"80";
        $res .= "\" rx=\"20";
        $res .= "\" ry=\"20";
        $res .= "\" fill=\"blue";
        $res .= "\" fill-opacity=\"0.5";
        $res .= "\" pointer-events=\"none";
        $res .= "\" stroke-width=\"2";
        $res .= "\" stroke=\"white";
        $res .= "\"/>\n";
        $pontos .= $res;
        $res  = "<text x=\"";
        $res .= $data[$i]["longitude"] - $longitude;
        $res .= "\" y=\"";
        $res .= $latitude - $data[$i]["latitude"]+60;
        $res .= "\" style=\"font-size:100;fill:rgb(0,0,255);text-anchor:middle";
        $res .= "\">";
        $res .= $data[$i]["title"];
        $res .= "</text>\n";
        $etiquetas .= $res;
}
echo "<g id=\"turismo\">\n";
echo $pontos;
echo "</g>\n";
```

**Figura 89 – Extracto da função que vai buscar as coordenadas à BD.**

## *5.2. Manipulação de Recursos Turísticos*

Ainda referente ao site do *turismonoave.com*, é dada a opção de um determinado recurso estar ou não georeferenciado. Isso implica que, estando georeferenciado um recurso, este exista na BD *MySQL* e também na *shapefile* dos recursos relativos ao concelho respectivo (com o mesmo ID), caso contrário, o recurso não pode existir na *shapefile*.

O que foi proposto fazer neste caso, foi criar um mecanismo que permitisse aos administradores de cada concelho manipular a informação relativa aos recursos turísticos desse mesmo concelho. Essa informação, como já se sabe, está disponível em *shapefiles*.

### 5.2.1. Estrutura de dados (*shapefiles*)

Uma *shapefile* de recursos turísticos de um determinado concelho é composta pela seguinte informação:

Data

☐ selected only ☐ within visible extent Selected count : 0

| UID | | NUM | TIPO | icon | DESIG | xdisplace | ydisplace | xanchor | yanchor | longitude | latitude |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ☐ | 1891 | 4.3.0 | 119.png | Pavilhão e Piscina do Gru... | 12 | 12 | 0 | 0 | -8,1655093... | 41,453803... |
| 2 | ☐ | 1892 | 4.2.0 | 93.png | Complexo Turístico de Ril... | 12 | 12 | 0 | 0 | -8,2148464... | 41,424854... |
| 3 | ☐ | 2592 | 2.2.0 | 228.png | O Povoado de Santo Ovídio | 12 | 12 | 0 | 0 | -8,1840618... | 41,455097... |
| 4 | ☐ | 2599 | 11.2.0 | 76.png | Auditório da Casa da Cult... | 12 | 12 | 0 | 0 | -8,1724996... | 41,450947... |
| 5 | ☐ | 2691 | 4.3.0 | 119.png | Complexo Desportivo Mu... | 12 | 12 | 0 | 0 | -8,1685055... | 41,447904... |
| 6 | ☐ | 1881 | 4.6.0 | 51.png | Restauradores da Granja | 12 | 12 | 0 | 0 | -8,1773717... | 41,447478... |
| 7 | ☐ | 1890 | 4.3.0 | 119.png | Pavilhão Gimnodesportiv... | 12 | 12 | 0 | 0 | -8,1715368... | 41,44969141 |
| 8 | ☐ | 1739 | 2.2.0 | 228.png | Igreja Matriz de Fafe | 12 | 12 | 0 | 0 | -8,1671746... | 41,454530... |
| 9 | ☐ | 3632 | 3.2.0 | 92.png | Aquário | 12 | 12 | 0 | 0 | -8,1793484... | 41,453505... |
| 10 | ☐ | 3638 | 3.2.0 | 92.png | Pinto da Costa | 12 | 12 | 0 | 0 | -8,1819054... | 41,453293... |
| 11 | ☐ | 1900 | 2.2.0 | 228.png | Ponte do Barroco / Ponte ... | 12 | 12 | 0 | 0 | -8,1856974... | 41,471595... |
| 12 | ☐ | 2941 | 2.4.0 | 14.png | Museu Hidroeléctrico de ... | 12 | 12 | 0 | 0 | -8,1837613... | 41,471953... |
| 13 | ☐ | 2880 | 1.4.1 | 110.png | Casa das Paredes | 12 | 12 | 0 | 0 | -8,1597938... | 41,465827... |
| 14 | ☐ | 3551 | 4.3.0 | 119.png | Barragem da Queimadela | 12 | 12 | 0 | 0 | -8,1636317... | 41,503458... |
| 15 | ☐ | 353 | 1.3.0 | 221.png | Parque de Campismo da ... | 12 | 12 | 0 | 0 | -8,1623119... | 41,504244... |
| 16 | ☐ | 1740 | 2.2.0 | 228.png | Igreja Nova de São José | 12 | 12 | 0 | 0 | -8,1687422... | 41,450110... |
| 17 | ☐ | 3639 | 3.2.0 | 92.png | Tónio Quim | 12 | 12 | 0 | 0 | -8,1573331... | 41,486042... |
| 18 | ☐ | 2882 | 1.4.0 | 90.png | Casa do Godim | 12 | 12 | 0 | 0 | -8,1714235... | 41,482714... |
| 19 | ☐ | 3167 | 1.1.1 | 202.png | Hotel Comfort Inn Fafe ** | 12 | 12 | 0 | 0 | -8,1757432... | 41,446170... |
| 20 | ☐ | 85 | 5.2.0 | 1.png | Albufeira de Queimadela | 12 | 12 | 0 | 0 | -8,1628623... | 41,506294... |
| 21 | ☐ | 2326 | 2.2.0 | 228.png | Igreja Românica de Arões | 12 | 12 | 0 | 0 | -8,2172251... | 41,456172... |

**Figura 90 – *Shapefile* de recursos turísticos.**

- **NUM –** A chave que o recurso tem na BD;
- **TIPO –** O tipo de recurso (ordem hierárquica dos recursos turísticos);
- **ICON –** Nome da imagem associada ao ícone;
- **DESIG –** Designação do recurso;
- **(X/Y)DISPLACE e (X/Y)ANCHOR –** Estes valores são usados no ficheiro SLD para o desenho da designação do recurso e representam a posição que o texto deve tomar relativamente ao ponto do recurso (ver especificação SLD). Estas variáveis são usadas para evitar sobreposição de texto em mapas;
- **LONGITUDE/LATITUDE –** Coordenadas do recurso.

### 5.2.2. Inserção de novos recursos turísticos

Quando um utilizador insere um novo recurso turístico e o quer georeferenciar, o processo interno no servidor consiste em primeiro guardar o recurso na BD e depois enviar um pedido ao servidor WFS-T para inserir um novo registo na *shapefile* correspondente. Um pedido típico de um *insert* a um servidor WFS-T seria:

```
function insert_pt_wfs($id, $nome, $lon, $lat, $icon, $concelho, $lplace){

$XPost = '<?xml version="1.0" encoding="latin1"?>';
$XPost .= "<wfs:Transaction service='WFS' version='1.0.0' ";
$XPost .= "xmlns:wfs='http://www.opengis.net/wfs' ";
$XPost .= "xmlns:topp='http://www.openplans.org/topp' ";
$XPost .= "xmlns:gml='http://www.opengis.net/gml' ";
$XPost .= "xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' ";
$XPost .= "xsi:schemaLocation='http://www.openplans.org/topp
http://localhost:8888/geoserver/wfs/DescribeFeatureType?typeName=topp:coord' >";
$XPost .= "<wfs:Insert>";
$XPost .= "<topp:".$concelho."recs>";
$XPost .= "<topp:the_geom>";
$XPost .= "<gml:Point srsName='http://www.opengis.net/gml/srs/epsg.xml#27354'>";
$XPost .= "<gml:coordinates decimal='.' cs=',' ts=' '>".$lon.",".$lat."</gml:coordinates>";
$XPost .= "</gml:Point>";
$XPost .= "</topp:the_geom>";
$XPost .= "<topp:DESIG>".$nome."</topp:DESIG>";
$XPost .= "<topp:icon>".$icon."</topp:icon>";
$XPost .= "<topp:NUM>".$id."</topp:NUM>";
$XPost .= "<topp:xdisplace>".$xdis."</topp:xdisplace>";
$XPost .= "<topp:ydisplace>".$ydis."</topp:ydisplace>";
$XPost .= "<topp:xanchor>".$xanc."</topp:xanchor>";
$XPost .= "<topp:yanchor>".$yanc."</topp:yanchor>";
$XPost .= "</topp:".$concelho."recs>";
$XPost .= "</wfs:Insert>";
$XPost .= "</wfs:Transaction>";
```

**Figura 91 – Código PHP para pedido de *insert* ao servidor WFS-T (1ª Parte).**

Nesta primeira parte constrói-se o pedido a fazer ao WFS. Pedido esse que segue a sintaxe de uma *transaction*, correspondente a um *insert*, sendo necessário identificar os campos e respectivos valores a inserir. Note-se que a etiqueta "<*topp:the_geom*>" refere-se à coluna *shape*, comum a todas as *shapefiles*, e onde é guardada a geometria da entidade.

```
$url = "http://localhost:8888/geoserver/wfs"; // enter the URL to post to here
$ch = curl_init();    // initialize curl handle
curl_setopt($ch, CURLOPT_URL,$url); // set url to post to
curl_setopt($ch, CURLOPT_RETURNTRANSFER,1); // return into a variable
curl_setopt($ch, CURLOPT_HEADER, 1);  // capture the returned headers
curl_setopt($ch, CURLOPT_TIMEOUT, 4); // times out after 4s
curl_setopt($ch, CURLOPT_POSTFIELDS, $XPost); // add POST fields
$result = curl_exec($ch); // run the whole process
// the headers returned from the server will now be stored in $result

settype($result,"string");
$pos = strpos($result, "SUCCESS");

if ($pos === false)
        return "ERRO";
else
        return "SUCESSO";

}
```

**Figura 92 - Código PHP para pedido de *insert* ao servidor WFS-T (2ª Parte).**

Numa segunda parte, cria-se uma instância da classe responsável por encaminhar o pedido, definem-se as opções correspondentes, envia-se o pedido e por fim testa-se se o resultado do pedido finalizou com sucesso ou deu erro.

**NOTA:** Para se poder enviar pedidos transaccionais ao servidor WFS usa-se o método POST do protocolo HTTP e, para isso, a opção mais simples é activar as extensões *curl* do PHP.

### 5.2.3. Actualização de recursos turísticos

Quando um utilizador faz uma actualização a um recurso georeferenciado, o processo é idêntico ao anterior mudando apenas o cabeçalho e a primeira parte da função, em que agora o pedido a fazer, dentro da operação *transaction* do WFS, é um *update* (ligeiramente diferente do pedido de *insert* dado que se tem de assegurar que o registo a alterar já existe).

```
function update_pt_wfs($id, $nome, $lon, $lat, $icon, $concelho, $lplace){

$XPost = '<?xml version="1.0" encoding="latin1"?>';
$XPost .= "<wfs:Transaction service='WFS' version='1.0.0' ";
$XPost .= "xmlns:topp='http://www.openplans.org/topp' ";
$XPost .= "xmlns:ogc='http://www.opengis.net/ogc' ";
$XPost .= "xmlns:gml='http://www.opengis.net/gml' ";
$XPost .= "xmlns:wfs='http://www.opengis.net/wfs'>";
$XPost .= "<wfs:Update typeName='topp:".$concelho."recs'>";
$XPost .= "<wfs:Property>";
$XPost .= "<wfs:Name>the_geom</wfs:Name>";
$XPost .= "<wfs:Value>";
$XPost .= "<gml:Point srsName='http://www.opengis.net/gml/srs/epsg.xml#27345'>";
$XPost .= "<gml:coordinates>".$lon.",".$lat."</gml:coordinates>";
$XPost .= "</gml:Point>";
$XPost .= "</wfs:Value>";
$XPost .= "</wfs:Property>";
$XPost .= "<wfs:Property>";
$XPost .= "<wfs:Name>DESIG</wfs:Name>";
$XPost .= "<wfs:Value>".$nome."</wfs:Value>";
$XPost .= "</wfs:Property>";
$XPost .= "<wfs:Property>";
$XPost .= "<wfs:Name>icon</wfs:Name>";
$XPost .= "<wfs:Value>".$icon."</wfs:Value>";
$XPost .= "</wfs:Property>";
$XPost .= "<wfs:Property>";
$XPost .= "<wfs:Name>xdisplace</wfs:Name>";
$XPost .= "<wfs:Value>".$xdis."</wfs:Value>";
$XPost .= "</wfs:Property>";
$XPost .= "<wfs:Property>";
$XPost .= "<wfs:Name>ydisplace</wfs:Name>";
$XPost .= "<wfs:Value>".$ydis."</wfs:Value>";
$XPost .= "</wfs:Property>";
$XPost .= "<wfs:Property>";
$XPost .= "<wfs:Name>xanchor</wfs:Name>";
$XPost .= "<wfs:Value>".$xanc."</wfs:Value>";
$XPost .= "</wfs:Property>";
$XPost .= "<wfs:Property>";
$XPost .= "<wfs:Name>yanchor</wfs:Name>";
$XPost .= "<wfs:Value>".$yanc."</wfs:Value>";
$XPost .= "</wfs:Property>";
$XPost .= "<ogc:Filter>";
$XPost .= "<ogc:PropertyIsEqualTo>";
$XPost .= "<ogc:PropertyName>topp:NUM</ogc:PropertyName>";
$XPost .= "<ogc:Literal>".$id."</ogc:Literal>";
$XPost .= "</ogc:PropertyIsEqualTo>";
$XPost .= "</ogc:Filter>";
$XPost .= "</wfs:Update>";
$XPost .= "</wfs:Transaction>";
```

**Figura 93 - Código PHP para pedido de *update* ao servidor WFS-T.**

### 5.2.3. Remoção de recursos turísticos

Também idêntico é o processo quando um utilizador pretende remover um registo georeferenciado, só mudando o cabeçalho da função e o pedido a fazer ao WFS.

```
function delete_pt_wfs($id, $concelho){

$XPost = '<?xml version="1.0" encoding="latin1"?>';
$XPost .= "<wfs:Transaction service='WFS' version='1.0.0' ";
$XPost .= "xmlns:cdf='http://www.opengis.net/cite/data' ";
$XPost .= "xmlns:ogc='http://www.opengis.net/ogc' ";
$XPost .= "xmlns:wfs='http://www.opengis.net/wfs' ";
$XPost .= "xmlns:topp='http://www.openplans.org/topp'>";
$XPost .= "<wfs:Delete typeName='topp:".$concelho."recs'>";
$XPost .= "<ogc:Filter>";
$XPost .= "<ogc:PropertyIsEqualTo>";
$XPost .= "<ogc:PropertyName>topp:NUM</ogc:PropertyName>";
$XPost .= "<ogc:Literal>".$id."</ogc:Literal>";
$XPost .= "</ogc:PropertyIsEqualTo>";
$XPost .= "</ogc:Filter>";
$XPost .= "</wfs:Delete>";
$XPost .= "</wfs:Transaction>";
```

**Figura 94 - Código PHP para pedido de *delete* ao servidor WFS-T.**

## 5.3. GeoWiki

Se se olhar para o fenómeno da Wikipédia pode-se considerar que existe uma maneira de tornar o processo de actualização de informação fácil e automático, sustentado pelos próprios utilizadores. Daí, pensou-se aplicar o mesmo princípio à IG. Os utilizadores da IG vão actualizando-a de forma natural e contínua, mas com a possibilidade de voltar a uma versão anterior. Procurou-se, então, desenhar um mecanismo que permita a edição de mapas pelos utilizadores, que contenha o mesmo controlo de versões da Wikipédia, como forma de controlo da actualização dos mapas. Com esta possibilidade, revoluciona-se toda a tradição de produção e manutenção de dados geográficos, já que os utilizadores, por exemplo, um grupo de caçadores, pode autonomamente produzir e manter informação geográfica através da Internet. O GeoWiki é a tecnologia que dá esta possibilidade.

Não é proposto, neste trabalho, implementar uma solução completa, mas apenas esquematizar um mecanismo possível, e mostrar que existe uma solução viável para este problema. Procurou-se também fomentar algum movimento para tornar este projecto completamente realizável.

Existem várias maneiras de aproximar este problema. Resolveu-se utilizar o projecto *Mapbuilder*, e uma base de dados *PostgreSQL* (com o módulo *Postgis*), para implementar esta solução. Assim, a maneira mais simples encontrada foi alterar o modelo de dados, na base de dados, e também alterar um par de ficheiros do núcleo do *Mapbuilder* de forma a conseguir a solução.

Este trabalho visa, como exemplo e dando continuidade aos exemplos deste capítulo, a rede viária do concelho da Trofa. Para tal, usam-se os *layers* do concelho e das estradas.

A solução que foi idealizada passa por ter uma tabela onde são guardadas todas as versões de todas as estradas (**estradas_hist**), uma tabela com a versão em que se está a trabalhar (**estradas**), uma tabela com informação relativa às várias

versões existentes (**versions**) e uma outra tabela que guarde informação de controlo (**vact**). Assim, sempre que se inicia a aplicação vai-se buscar a versão actual e "enche-se" a tabela **estradas** com informação vinda de **estadas_hist**, sempre que se gravam as alterações é gerada um nova versão e copia-se todo o conteúdo de **estradas** para **estradas_hist**. Desta maneira, pode-se sempre ir buscar uma versão anterior qualquer à tabela **estradas_hist** e trabalhar sobre essa versão.

### 5.3.1. Modelo de dados

Guardadas as entidades relativas ao concelho e às estradas da Trofa na base de dados em forma de tabelas, vai-se alterar a tabela relativa às estradas. Esta tabela tem como colunas o identificador da linha (**gid**), a geometria (normalmente **the_geom**), e todos os atributos da *layer*. O objectivo é fazer desta tabela o repositório de todas as versões das estradas.

O primeiro passo é mudar o nome desta tabela para explicitar que esta se torna o histórico.

```
ALTER TABLE estradas
RENAME TO estradas_his
```

**Figura 95 – SQL para renomeação da tabela *estradas*.**

Seguidamente é necessário fazer mais alterações nesta tabela. É preciso criar mais um campo para a solução proposta: a versão (**version**) e também alterar a chave primária de **gid** para o par (**version**, **gid**).

```
ALTER TABLE estradas_hist
  ADD version int4 NOT NULL DEFAULT 1,
  DROP CONSTRAINT estradas_pkey,
  ADD CONSTRAINT vr_pkey PRIMARY KEY (version,gid);
```

**Figura 96 – SQL para alteração da tabela *estradas_hist*.**

O próximo passo é criar uma tabela que irá conter a versão actual das estradas. Esta tabela nova é baseada na tabela anterior (**estradas_hist**) com a excepção de que não será necessário guardar a versão. É necessário criar também um campo nesta nova tabela que guarde o identificador de cada estrada (registo) da tabela de histórico, mas que não seja chave na nova tabela. Isso acontece porque, aquando da criação de *FeatureTypes* no *GeoServer*, a chave das tabelas não fica acessível de uma maneira simples como os outros campos da tabela, e essa é precisa para as operações de *update* e *delete*.

```
CREATE TABLE estradas
(
  gid int4 NOT NULL DEFAULT nextval('estwgs84_gid_seq'::regclass),
  entity varchar(16),
  handle varchar(16),
  layer varchar(254),
  color int4,
  linetype varchar(254),
  elevation numeric,
  thickness numeric,
  text_ varchar(254),
  shape_leng numeric,
  the_geom geometry,
  estid int4 NOT NULL DEFAULT 0,
  CONSTRAINT gid_pk PRIMARY KEY (gid),
  CONSTRAINT enforce_dims_the_geom
    CHECK (ndims(the_geom) = 2),
  CONSTRAINT enforce_geotype_the_geom
    CHECK (geometrytype(the_geom) = 'MULTILINESTRING'::text
    OR the_geom IS NULL),
  CONSTRAINT enforce_srid_the_geom CHECK (srid(the_geom) = -1)
 )
```

**Figura 97 – SQL para alteração da tabela *estradas*.**

Vai-se também criar seguidamente uma tabela com informação sobre as versões existentes.

```
CREATE TABLE versions
(
  vid int4 NOT NULL DEFAULT 1,
  vuser varchar(16) NOT NULL DEFAULT 'nasf'::character varying,
  vdate timestamp NOT NULL DEFAULT now(),
    CONSTRAINT vid_pk PRIMARY KEY (vid)
)
```

**Figura 98 – SQL para criação da tabela *versions*.**

Finalmente, cria-se uma tabela de controlo para a aplicação. Nesta tabela, guarda-se a versão actual (**vid**), um campo para controlar se a versão em que estamos a trabalhar está actualizada (**is_upd**) e o valor máximo usado nos identificadores de linha, identificador de cada estrada, da tabela **estradas_hist** (**maxestid**).

```
CREATE TABLE vact
(
  vid int4 NOT NULL,
  is_upd bool NOT NULL DEFAULT true,
  maxestid int4 NOT NULL DEFAULT 0,
  CONSTRAINT vidk PRIMARY KEY (vid),
  CONSTRAINT vid_fk FOREIGN KEY (vid)
    REFERENCES versions (vid) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

**Figura 99 – SQL para criação da tabela *vact*.**

### 5.3.2. Aplicação

Para demonstrar o funcionamento da base de dados com suporte de histórico, foi criada uma aplicação com o *Mapbuilder*, na qual é possível alterar um mapa criando uma nova versão do mesmo, bem como navegar num histórico de versões (assumindo uma qualquer versão anterior).

Nessa aplicação é possível visualizar algumas camadas relativas ao mapa da Trofa:

- Freguesias;
- Estradas.

Destas, as operações serão feitas sobre a camada relativa às estradas, sobre a qual se pode efectuar as seguintes operações:

- Inserir uma estrada;
- Actualizar uma estrada;
- Remover uma estrada;
- Informação sobre cada estrada (entidade).

Para além disso, pode-se ter acesso a operações de:

- Zoom;
- *Pan*;
- *Reset*;
- Guardar a versão actual;
- Saltar para uma versão anterior.

#### 5.3.3.1. Configuração

Para construir o GeoWiki, na solução proposta, precisa-se de um servidor WMS e WFS-T e para isso resolveu-se, mais uma vez recorrer ao *GeoServer*. É

necessário também um cliente *Web Mapping*, a opção recaiu sobre o *Mapbuilder*, e de um servidor *Web* para as *scripts* PHP, o suporte de toda a aplicação.

#### *5.3.3.1.1. GeoServer*

Na secção 4.2.1. viu-se, em pormenor, como se configura o *GeoServer* para guardar IG. Assim, para este caso, foi criado um *DataStore* (**trofa**) para a ligação aos dados em *PostGIS* e dois *FeatureTypes* para mostrar os dados das freguesias e estradas (respectivamente, **freguesias** e **estradas**).

O *DataStore* terá como *namespace* o prefixo **topp**, pois, como se pretende fazer pedidos de *insert*, *update* e *delete* ao WFS, este *namespace* é o único que possibilita essas operações (é um *bug* do *GeoServer* que até à data de início desta aplicação ainda não tinha sido corrigido).

Para os *FeatureTypes* usa-se o sistema de coordenadas WGS84 (SRS 4326, o que implica que a IG na base de dados esteja nesse sistema, dado que o *Mapbuilder*, por defeito, não traz suporte ao sistema de coordenadas *Datum* 73) e os estilos *green* e *simple_roads* (para **freguesias** e **estradas**, respectivamente).

#### *5.3.3.1.2. Mapbuilder*

Na secção 4.3.3.1. viu-se, em pormenor, como se configura o *Mapbuilder* para mostrar IG. Assim, criou-se um ficheiro "*trofa.xml"* com os dois *layers* que vão ser mostrados (**topp:freguesias** e **topp:estradas**). O *layer* **topp:estradas** tem de ter a opção de inquérito activada (atributo *queriable*="1") para que seja possível mostrar a informação de uma estrada escolhida.

```
<Layer queryable="1" hidden="0">
    <Server service="OGC:WMS" version="1.1.1" title="Layers">
            <OnlineResource xlink:type="simple"
                xlink:href="http://localhost:8888/GeoServer/wms"/>
    </Server>
    <Name>topp:estradas</Name>
        …
  </Layer>
```

**Figura 100 – Extracto da definição de um *Layer*.**

No ficheiro de configuração do *Mapbuilder* ("*config.xml*") activam-se os *widgets* para *zoom*, *pan*, *editline* e *getfeature*.

Para a implementação das operações de manipulação de *features* (*insert*, *update* e *delete*), verificou-se que o *Mapbuilder* utiliza o seguinte mecanismo: existe um ficheiro XML de entradas com as especificações da entidade pretendida, que é submetido a um XSL que converte as especificações no pedido respectivo a fazer ao WFS-T.

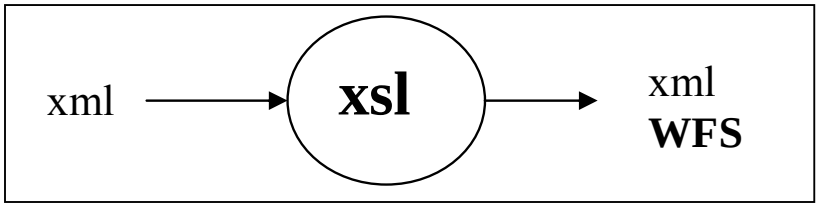



**Figura 101 – Implementação das operações de manipulação de entidades.**

Portanto, existe um ficheiro XML de entrada que funciona como *template* e contém, no neste caso, a estrutura de como é que está guardada/acessível a informação de uma estrada na base de dados. O ficheiro é comum aos três pedidos (*insert*, *update* e *delete*), é incluído no ficheiro de configuração do *Mapbuilder* (no *widget editline*) e apresenta o seguinte formato:

```
<?xml version="1.0" encoding="UTF-8"?>
<gml:featureMember xmlns:topp="http://www.openplans.org/topp"
xmlns:gml="http://www.opengis.net/gml">
  <topp:estradas gid="estradas.1">
    <topp:the_geom>
      <gml:MultiLineString srsName="epsg:4326">
        <gml:lineStringMember>
          <gml:LineString>
            <gml:coordinates decimal="." cs="," ts=" "></gml:coordinates>
          </gml:LineString>
        </gml:lineStringMember>
      </gml:MultiLineString>
    </topp:the_geom>
    <topp:entity>Polyline</topp:entity>
    <topp:linetype>Continuous</topp:linetype>
    <topp:text_>Estrada ...</topp:text_>
  </topp:estradas>
</gml:featureMember>
```

**Figura 102 – Ficheiro *template* usado para geração de pedidos ao WFS-T,**

A criação do XSL é feita para que, quando for aplicado ao XML de entrada, produza o XML com o pedido correcto correspondente à operação pretendida. Pedido esse que é depois enviado ao servidor.

Todo este processo é executado de uma forma automática, mas que ainda não está bem implementado para todas as operações de manipulação. Portanto, para esta aplicação aproveitou-se só parte deste processo, ou seja, aproveitou-se o facto de existir um ficheiro *template* para gerar o formulário para as operações de manipulação e depois o pedido é gerado e enviado ao WFS-T por PHP. Assim, tem de se fazer uma pequena alteração ao XSL (*FeatureList.xsl*) que processa o ficheiro XML de entrada (*template_estradas.xml*). Alteração essa que passa por adicionar um *form* HTML para cada pedido disponibilizado.

```
…
<div>
  <h3>Feature List</h3>
  <form method="post" action="GeoWiki.php">
    <xsl:apply-templates/>
    <input type="submit" name="Insert" value="Insert"/>
    <input type="submit" name="Update" value="Update"/>
    <input type="submit" name="Delete" value="Delete"/>
  </form>
</div>
...
```

**Figura 103 – Código para os pedidos ao WFS-T.**

No ficheiro de apresentação do *Mapbuilder* (*index.html*) criaram-se as secções para apresentação do mapa com as freguesias e estradas, legendas e o formulário de edição (associado à *FeatureList*). Este ficheiro será incluído na *script* PHP de apresentação.

#### *5.3.3.1.3. Servidor Apache*

No servidor apache é necessário activar as seguintes extensões (ficheiro *php.ini*):

- **php_pgsql –** Para acesso às funções PHP que operam sobre bases de dados *PostgreSQL*;
- **php_curl –** Para permitir enviar pedidos em XML ao servidor WFS.

### 5.3.3.2. Funcionamento

O GeoWiki pode ser executado em qualquer browser que suporte o *Mapbuilder*. Uma *script* PHP é responsável por todo o funcionamento e controlo da aplicação.

Aquando do arranque, a aplicação vai buscar a versão actual (tabela **vact**), limpa o conteúdo da tabela que é usada para trabalhar com a versão actual da IG (tabela **estradas**) e volta a preencher esta tabela com os dados vindos da tabela de histórico (tabela **estradas_hist**) que correspondem à versão actual. Seguidamente actualiza o valor do último registo inserido e marca a versão corrente como estando actualizada (campos da tabela **vact**).

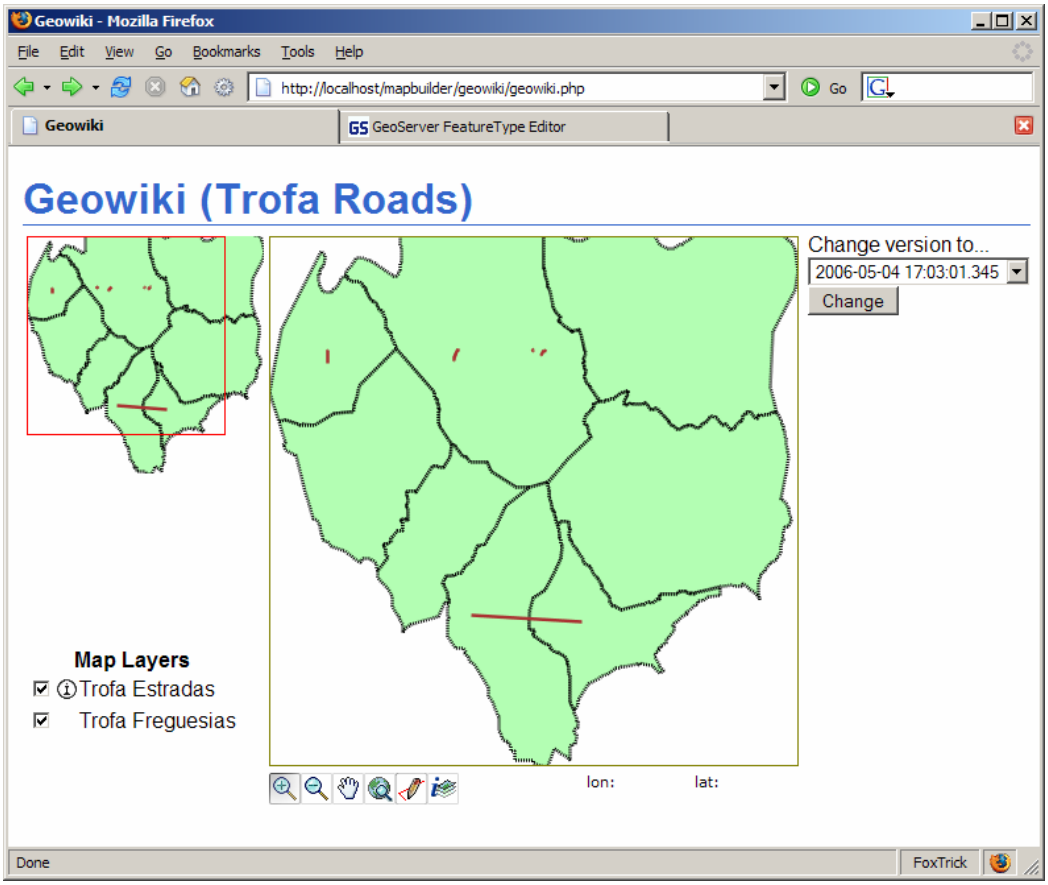



**Figura 104 – Inicialização do GeoWiki.**

Aquando de uma operação de manipulação de IG, a aplicação gera um pedido XML baseado no formulário e na operação correspondente para enviar ao WFS. Se o pedido for válido, e correctamente executado, vai actualizar a IG na tabela que guarda a versão actual e a versão actual é marcada como não estando actualizada. Assim, vai aparecer uma nova opção que permite gravar as alterações efectuadas.

Quando o utilizador escolhe gravar as alterações, a aplicação gera uma nova versão com a data actual, copia todos os registos da tabela **estradas** para a tabela de histórico em que esses registos são marcados com o identificador da nova versão e actualiza a tabela de controlo com os novos valores da nova versão, último registo inserido e marca a nova versão como actualizada.

Quando o utilizador escolher regressar a uma versão anterior, a aplicação actualiza os valores da tabela de controlo (a versão actual passa a ser a escolhida pelo utilizador e é marcada como actualizada), a tabela **estradas** é limpa e preenchida com os registos vindos da tabela de histórico referentes à versão escolhida.

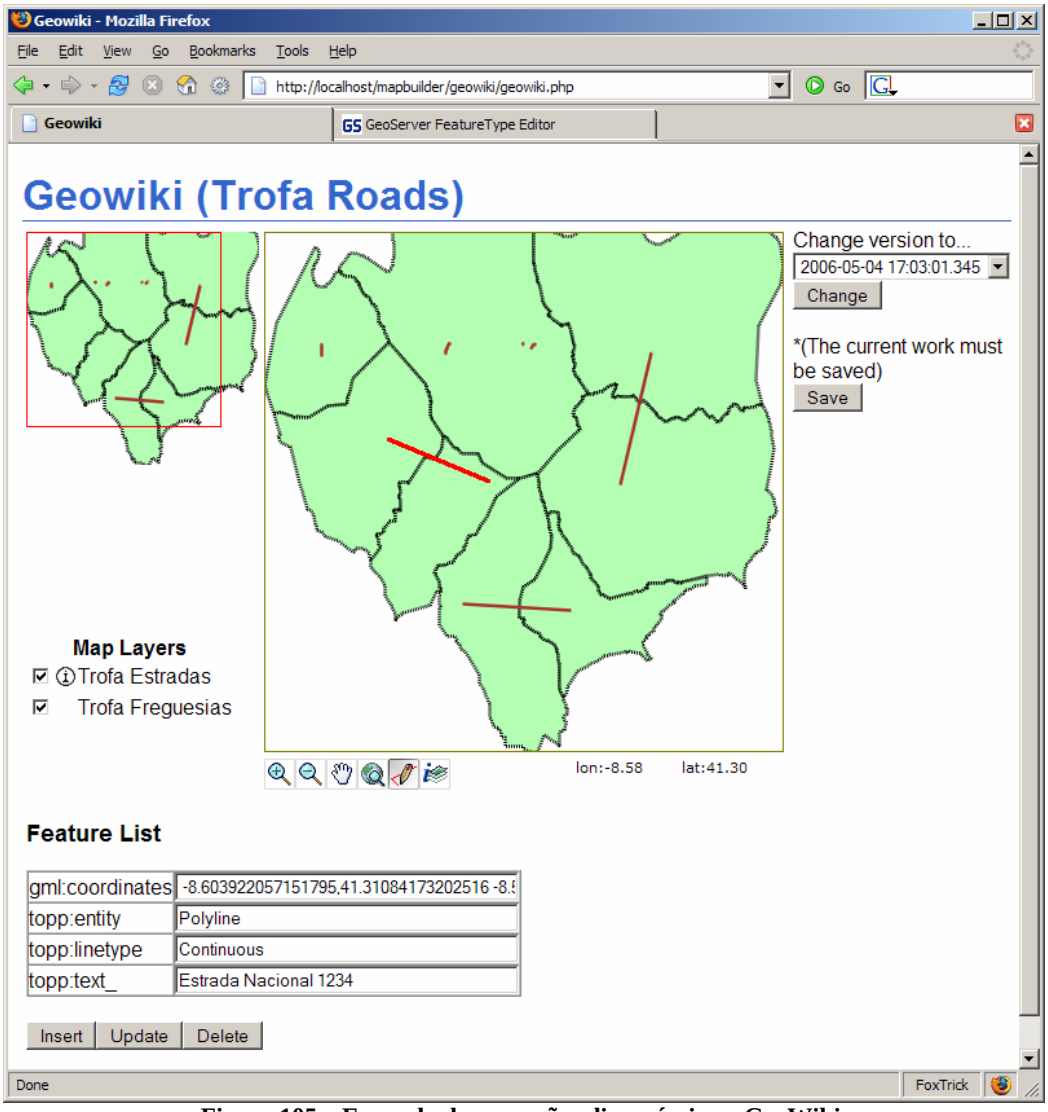



**Figura 105 – Exemplo de operações disponíveis no GeoWiki.**

# 6. Conclusões e Trabalho Futuro

Este capítulo encerra esta dissertação e nele se discutem a maneira como foram atingidos os objectivos propostos, ideias para dar continuidade à vertente de actualização de IG na *Web* (GeoWiki, neste caso) e considerações finais que se consideram oportunas com a conclusão deste trabalho de mestrado.

## *6.1. Objectivos atingidos*

Relembrando os objectivos propostos, na secção 1.2, tem-se a tecer os seguintes comentários:

**Compreender as especificidades da IG.**

Este objectivo, essencial para a introdução à temática da IG tratada em computador, foi o primeiro a ser cumprido. Isto porque a IG tem de ser tratada de uma forma particular no contexto das Ciências da Computação, pois tem características próprias [Rocha05]. No capítulo 2, começa-se por identificar estas características e, de seguida, são percorridas algumas representações em computador, já que nesta dissertação foi desenvolvido um protótipo onde se quebra a forma tradicional de guardar a IG (no GeoWiki, são guardadas as múltiplas representações de cada entidade que esta vai tendo ao longo do tempo).

Define-se IG como um tuplo que associa uma série de atributos à sua respectiva localização. Foram apresentadas uma série de ciências (ditas de geociências) que contribuem para o tratamento e aquisição de IG, para posterior utilização em computador. Assim, é apresentado o conceito de SIG que são sistemas computacionais onde é armazenada e manipulada IG.

Mostrou-se também o que é que está disponível no "mercado" para representar as especificidades da IG de um modo vectorial (contrapondo com a representação matricial que foge do foco desta dissertação). Desta forma, as entidades geográficas do mundo real são modeladas essencialmente por: pontos, linhas e polígonos. Essas entidades podem ser representadas em: SGBD (através de uma especificação criada para o efeito - SFS para SQL – à qual muitas companhias aderiram, sejam livres ou proprietárias), ficheiros e estruturas de dados proprietárias (como são os casos das *shapefiles* e *Geodatabases* da ESRI) ou através de GML (especificação baseada em XML criada para representar IG).

O GML é mais indicado para distribuir IG[46] enquanto que os SGBD são a melhor opção para guardar grandes volumes desta.

**Conhecer e desenvolver soluções que manipulem a IG num ambiente *Web*.**

Estes objectivos foram cumpridos e estão descritos nos capítulos 3, 4 e 5.

[46] A agência inglesa de cartografia *Ordenance Survey* já só distribui a principal cartografia produzida em GML.

A Internet trouxe muitas vantagens para a disseminação de IG, nomeadamente no que diz respeito à acessibilidade (mais gente a aceder à IG e a várias fontes), facilidade de actualização (servidores centralizados para todos os utilizadores actualizarem a IG) e como meio de comunicação (conjuntos de dados geográficos são transaccionados via Internet).

O consórcio OGC trabalha com o objectivo de criar mecanismos e regras para os utilizadores manipularem IG de um modo uniforme e coerente, através de formatos abertos. O trabalho do OCG é enquadrado pelo *OGC Reference Model* que preconiza uma teia de Geo-WS. Dentro destes, são destacados: o serviço de mapas (WMS, para disponibilizar mapas) e o serviço de entidades (WFS-T, para consultar e alterar entidades geográficas). Relativamente ao WMS, o OGC criou uma forma de apresentar a informação contida dos mapas disponibilizados (através da linguagem SLD) e permite também guardar determinados contextos de apresentação definidos pelos utilizadores (através de documentos WMC).

Para explorar os Geo-WS foram estudadas e testadas duas soluções *open source* que implementam os serviços preconizados pelo OGC: os projectos *deegree* e *GeoServer*. Mas também com a consciência de que há outras formas de manipular IG via *Web*, mostra-se como pode ser feita através de SVG e usando a API do *Google Maps* (esta última não permite actualizar IG, apenas permite visualizar).

O *site turismonoave.com* e o GeoWiki foram as soluções desenvolvidas com o objectivo de testar e provar as teorias defendidas nesta dissertação, nomeadamente de que é possível criar soluções que manipulem IG em ambiente *Web* de simples utilização para todos os utilizadores de IG (quer sejam conhecedores do meio ou não), baseadas em soluções *open source* e sem recorrer a extensões aos navegadores vulgares (IE e *Mozilla*).

**Contribuir para facilitar a apresentação e manipulação da IG na *Web*.**

Devido aos conhecimentos adquiridos no estudo e resultante da exploração das ferramentas existentes, pode-se concluir que já existe tecnologia, conhecimentos e vontade para criar aplicações que facilitem a apresentação e manipulação de IG na *Web* (e sem necessidade de grandes investimentos). Pode-se afirmar isso com toda a convicção e a prova está no contributo com o *site turismonoave.com* e o GeoWiki. Se não vejamos:

O *site turismonoave.com* foi construído em PHP e a BD de suporte está em *MySQL*, tudo tecnologias livres. Para o SIG, optou-se por recorrer aos Geo-WS, implementando um WFS-T, que permite a actualização da IG[47]. Para implementar o WFS-T, optou-se pelo *GeoServer*, embora também fosse utilizado o *deegree* em muitas experiências. Os dados geográficos estão representados em *shapefiles* devido ao levantamento feito com as ferramentas da ESRI. Para visualizar os mapas foi criada uma *script* PHP que lança o pedido (baseado nas coordenadas do recurso escolhido) ao servidor WMS e mostra a imagem retornada por este. Também para a visualização da IG, foi integrada no site uma outra *script* PHP, que usa a API do

[47] Inicialmente usamos SVG para facilitar a determinação das coordenadas de cada novo recurso, e todas as actualização (inserção e alteração de coordenadas) eram feitas por scripts PHP.

*Google Maps* para mostrar mapas de satélite. A actualização da IG é feita usando formulários e é actualizada na BD e nas *shapefiles.*

Com esta abordagem, criou-se uma total independência entre a componente de IG e o próprio site; toda a comunicação entre as partes é feita através de pedidos ao WFS-T. Isto permite, por exemplo, que se desenvolvam outras aplicações que interajam com o WFS-T, usufruindo da IG sempre actualizada, sem necessidade de réplicas e mecanismos mais ou menos sofisticados de sincronização.

O GeoWiki foi desenvolvido sobre três projectos *open source*: o *Mapbuilder*, o *GeoServer* e o SGBD *Postgres* (com *PostGIS*). O *Mapbuilder* é um cliente para serviços WMS e WFS-T. É muito estável, completo e rápido (faz uso da tecnologia AJAX, portanto a maioria das acções são feitas na máquina do utilizador e não no servidor). Apenas foi preciso fazer algumas pequenas alterações à distribuição do *Mapbuilder*, para ajustar os pedidos a fazer ao WFS-T à estrutura de dados usada no exemplo. Os dados geográficos estão representados em *PostGIS*, o que também contribui para uma boa performance, quer do *GeoServer* quer do *Mapbuilder*. O controlo de toda a aplicação é feito em PHP.

Parece-me que este GeoWiki só por si sustenta a tese de que é possível manipular IG sobre a *Web*, e de uma forma controlada, recorrendo a mecanismos de gestão de versões entidade a entidade.

## *6.2. Trabalho futuro*

Não há dúvidas nenhumas que continuar a acompanhar os desenvolvimentos do OGC (nomeadamente na evolução dos serviços já existentes para a fase 3 ou com o aparecimento de novos serviços), bem como perceber os movimentos das companhias produtoras de serviços e software de IG é importante para antecipar o que irá ser cada vez mais relevante nesta área.

Em termos genéricos, continuar a participação em projectos com o objectivo geral de promover a massificação da IG, a interoperacionalidade entre os sistemas que a suportam e a disponibilização e manipulação da IG em ambiente *Web* vai continuar a ser, na realidade, uma área em que investir, pois continua a ser uma exigência dos utilizadores.

Como o *site turismonoave.com* é um projecto proprietário que cumpriu, e até excedeu, as expectativas e requisitos em termos de disponibilização e manipulação de IG, qualquer trabalho futuro passaria sempre pela aprovação de novos requisitos por parte da entidade que gere o projecto. No entanto, penso que se poderia melhorar a performance do site, se as entidades geográficas estivessem guardadas no mesmo SGBD que o resto da informação do site (tudo em *MySQL* com extensões GIS ou em *PostGIS* em vez do actual formato misto *MySQL* e *shapefiles*) e se fosse integrado o *MapBuilder* na página de apresentação da IG. Este projecto, serviu essencialmente para ensaiar uma variedade de aproximações à disponibilização de IG na *Web*. No projecto posterior, o GeoWiki, aproveitou-se muita desta experiência.

Relativamente ao GeoWiki, é um projecto científico mais ambicioso e que serviu, para já, mais como prova de conceito do que como solução final. Para além de um melhor controlo e automatização das acções, é necessário adaptar o GeoWiki para diferentes contextos geográficos (não só para rede viária). A sua prova de fogo seria perante a sua massiva utilização e aí teriam-se respostas para as questões: A construção colaborativa de conjuntos de dados espaciais é uma alternativa ao processo tradicional? Conseguem-se concretizar políticas de resolução de conflitos entre contribuições? A qualidade da informação resultante é cada vez melhor? Essas respostas poderão, ou não, levar a uma alteração significativa do actual modo de funcionamento do GeoWiki.

## *6.3. Considerações finais*

Partiu-se com o objectivo de tornar a IG tão facilmente actualizável via *Web*, como já o é a informação mais textual (ou estruturada, como acontece com as tradicionais bases de dados).

Claro que a construção colaborativa de dados geográficos está sujeita às mesmas premissas do restante tipo de informação actualizável via *Web*: quem pode actualizar, qual a qualidade dos contributos, como se pode entusiasmar os utilizadores a colaborarem, etc. Contudo, a IG levanta problemas relacionados com a sua complexidade intrínseca. Isto obriga a modelos de representação mais complexos e a formas de interacção e visualização expeditas. Estas questões obrigam a ter uma camada computacional mais poderosa, que poderá estar encapsulada em Geo-WS, como defendemos neste trabalho.

Os Geo-WS, como já se sabe, encapsulam, manipulam e disponibilizam grandes quantidades de IG. A sua completa integração com as tecnologias XML é claramente uma vantagem que devemos explorar.

O desenvolvimento de um projecto desta natureza passa por:

- Fase 3, da iniciativa Geo-WS do OGC;
- Apostar em formatos abertos; se estiverem codificados em XML, tem-se a vantagem de herdar uma data de ferramentas já existentes;
- Apostar numa plataforma de Geo-WS.

Para finalizar esta dissertação, resta dizer que esta está englobada numa série de iniciativas e trabalhos que originaram outras tantas dissertações de mestrado e doutoramento em IG, levadas a cabo no laboratório de SIG do Departamento de Informática da Universidade do Minho. Assim, destacam-se até ao momento, o trabalho de dissertação de doutoramento de Jorge Rocha (Informação Geográfica: Meta-Informação, Codificação e Visualização) e as dissertações de mestrado de André Araújo (Web Services na Informação Geográfica) e Ricardo Martins (Exploração do GML e Web Services para uso de SIG em PDA).

# Bibliografia

[Araújo05] Araújo, Mário André, 2005. Web Services na Informação Geográfica. Dissertação de Mestrado, Universidade do Minho.

[Bray00] Bray, T.; Paoli, J.; Sperberg-MacQueen, C. M., Maler, Extensible markup language 1.0 . 2. ed. World Wide Web Recommendation. [S.I.]: W3Consortium, 2000.

[Buehleretal96] Buehler K et al. The OpenGIS guide: Introduction to interoperable geoprocessing. Technical Report, Open GIS Consortium, Inc., 1996.

[Câmaraetal04] Câmara, Gilberto; Monteiro, Antônio Miguel; Medeiros, José Simeao de. 2004. Introdução à Ciência da Geoinformação. São José dos Campos, INPE, 2004.

[Caviedes05] Caviedes, Pedro Grafulic, 2005. GEODESIA TEORIA Y PRACTICA. Universidad de Santiago de Chile.

[Clementinietal93] Clementini, E., Di Felice, P., & van Oosterom, P., 1993. A small set of formal topological relationships suitable for end-user interaction. In D. Abel, & B. C. Ooi (Eds.), Advances in spatial databases - Third International Symposium, SSD'93, Singapore (pp. 277-295). Berlin: Springer.

[Costa02] Costa, G., O Modelo de Web Services — Como Desenvolver Aplicações em uma Nova Arquitetura de Software, *Promon Business & Technology Review Series*, n.4, 2002.

[Coxetal03] COX, S.; DAISEY, P.; LAKE, R.; PORTELE, C.; WHITESIDE, A. (ed), 2003. OpenGIS® Geography Markup Language (GML) Implementation Specification. Open Geospatial Consortium, Inc.

[Craigetal02] Craig, J.W.; Harris, T.M. & Weiner, D. (2002) Community participation and Geographic Information Systems. Taylor and Francis. London and New York.

[Çöltekinetal97] Çöltekin, Arzu; Çöltekin, Çağrı; Hatem,Onur Volkan; Vural, Arife, Berlin 1997. INTERNET AND GIS.

[Davis03] Davis, David E. 2003. GIS for Everyone: Exploring Your Neighborhood and Your World with a Geographic Information System, 3rd Edition. ESRI Press.

[Dogruetal04] Dogru, A. Garagon; Selcuk, T.; Ozener, H.; Gurkan, O.; Toz, G., 2004. DEVELOPING A WEB-BASED GIS APPLICATION FOR EARTHQUAKE INFORMATION

[ESRI98] ESRI Shapefile Technical Description, An ESRI White Paper—July 1998

[ESRI00] Esri Environmental Systems Research Institute Inc. (2000). Object Modelling and Geodatabases. ArcInfo 8.0 Pre-Release. 2000.

[Faria02] Faria, Nuno. 2002. Visualizador de mapas para PDAs em SVG. Technical report, Universidade do Minho. Relatório de Estágio da Licenciatura de Matemática e Ciências da Computação.

[Gemael99] Gemael, C. 1999. Geodésia Física, Editora da UFPR, Curitiba PR 1999.

[Gomesetal06] Gomes, Manuel Mesquita T. F; Rocha, Artur Jorge da Silva; Coelho, António Fernando; Sousa, A. Augusto Sousa. Acesso Interoperável a Informação Geográfica para Disponibilização de Modelos Urbanos 3D em Dispositivos Móveis. Xata 2006.

[IbmDb2Ref] IBM DB2 Spatial Extender - Referência e Manual do Usuário Versão 8.

[Kingstonetal03] Kingston, R.; Evans, A & Carver, S. Public participation via On-line Democracy (2003). *In* Planning Support Systems in Practice. Edit. Geertman, S. & Stillwell, J. Springer, New York.

[Klosterman01] Klosterman, R. E. (2001) Planning Support Systems: A New perspective on Computeraided Planning. *In* Planning Support Systems: Integrating Geographic Information Systems, Models, and Visualization Tools, Edit. R. K. Brail e R.E. Klosterman, ESRI Press. California.

[Lassen98] LASSEN, A. R. O., J.; OSTERBYE, K., 1998, Object Relational Modeling, Centre for Object Technology (COT).

[Martins04] Martins, Ricardo Alexandre G. C., 2004. Exploração do GML e Web Services para uso de SIG em PDA's. Dissertação de Mestrado, Universidade do Minho.

[Martinsetal03] Martins, Ricardo, Jorge Gustavo Rocha, and Pedro Henriques. 2003b. Segurança dos web services no comércio electrónico móvel. In coopmedia 2003 - Workshop de Sistemas de Informação Multimédia, Cooperativos e Distribuídos, Porto, Outubro.

[MüllerGarcia01] Müller, Mauricio; Garcia, Karina Sanches. 2001. Modelos Digitais de Terreno em escalas globais e sua utilização em recursos hídricos. Lactec – Instituto de Tecnologia para o Desenvolvimento.

[Murray03] MURRAY, C., 2003, Oracle® Spatial User's Guide and Reference 10g Release 1(10.1), Redwood City, Oracle Corporation, p. 602.

[MySqlMan06] MySQL 5.0 Reference Manual, 2006-06-13 (revision: 2366)

[OGC99] OGC, 1999. The OpenGIS Abstract Specification – Topic 6: The Coverage Type and its Subtypes. Open Geospatial Consortium, Inc.

[OGC99049] Open GIS Consortium, Inc. OpenGIS; Simple Features Specification For SQL Revision 1.1 - OpenGIS Project Document 99-049, Release Date: May 5, 1999

[OGCRM03] Percivall, George, 2003. OGC Reference Model. Open Geospatial Consortium Inc.

[OGCSLD02] Open GIS Consortium Inc., 2002. Styled Layer Descriptor Implementation Specification Version 1.0.0.

[OGCWFS02] Open GIS Consortium Inc., 2002. Web Feature Service Implementation Specification Version 1.0.0.

[OGCWMC03] Open GIS Consortium Inc., 2003. Web Map Context Implementation Specification Version 1.0.0.

[OGCWMS01] Open GIS Consortium Inc., 2001. Web Map Service Implementation Specification Version 1.1.1.

[PengBeimborn98] Peng, Zhong-Ren; Beimborn, Edward A., 1998. Internet GIS and Its Applications in Transportation.

[Percivall03] Percivall, G. (ed), 2003. OpenGIS® Reference Model. Document number OGC 03-040 Version: 0.1.3. Open Geospatial Consortium, Inc.

[Rocha03] Rocha, Noé Amorim. 2003. Visualizador de mapas SVG para PDAs. Technical report, Universidade do Minho. Relatório de Estágio da Licenciatura de Matemática e Ciências da Computação.

[Rocha03] Rocha, Jorge Gustavo. 2003. A java based svg viewer for the pocket pc. In SVGOpen 2003 - 2nd Annual Conference on Scalable Vector Graphics, Vancouver, Julho.

[Rocha05] Rocha, Jorge Gustavo, 2005. Informação Geográfica: Meta-Informação, Codificação e Visualização.

[Sonnet04] Sonnet, J. (ed), 2004. OWS 2 Common Architecture: WSDL SOAP UDDI. Discussion Paper OGC 04-060r1, Version: 1.0.0. Open Geospatial Consortium, Inc.

[Temba00] Temba, Plínio. 2000. Fundamentos da Fotogrametria. Departamento de Cartografia, UFMG.

[UchoaFerreira04] Uchoa, Helton Nogueira & Ferreira, Paulo Roberto - Geoprocessamento com Software Livre, (versão 1.0) 26102004.

[Urman02] URMAN, S. Oracle 9i Programacao PL/SqL. Rio de Janeiro: Editora Campos,2002. 552 p.